NBSIR 75-924

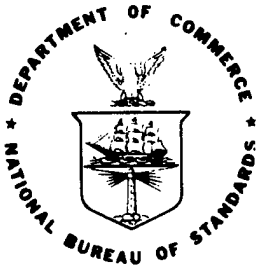
Evaluation, Revision and Application of the NBS Stylus/ Computer System for Surface Roughness Measurement: Minicomputer Software

E. Clayton Teague

Institute for Basic Standards
National Bureau of Standards
Washington, D. C. 20234

April 1975

Final Report



U S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

NBSIR 75-924

EVALUATION, REVISION AND APPLICATION OF THE NBS STYLUS/ COMPUTER SYSTEM FOR SURFACE ROUGHNESS MEASUREMENT: MINICOMPUTER SOFTWARE

E. Clayton Teague

Institute for Basic Standards
National Bureau of Standards
Washington, D. C. 20234

April 1975

Final Report

U.S. DEPARTMENT OF COMMERCE, Rogers C.B. Morton, ***Secretary***
James A. Baker, III, ***Under Secretary***
Dr. Betsy Ancker-Johnson, ***Assistant Secretary for Science and Technology***

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, ***Acting Director***

Table of Contents

List of Figures

MINICOMPUTER SOFTWARE FOR THE NBS STYLUS/COMPUTER SYSTEM FOR SURFACE ROUGHNESS MEASUREMENTS

1. Introduction

A thorough description of all the software used at NBS for characterizing surface texture is given in this report. The description includes flow diagrams and detailed, annotated listings of machine language programs for step-calibrating the system, for acquiring digitized surface profiles and for calculating from these profiles important parameters and statistical functions. Parameters and functions included are the arithmetic average value, mean square value, average wavelength, average slope, amplitude density function and autocorrelation function.

The report was originally written as an appendix for the NBS Technical Note, "Evaluation, Revision and Application of the NBS Stylus/Computer System for Surface Roughness Measurement." However, due to its size and specialized content, the appendix was issued as this separate volume.

This appendix contains the following information about the computer software used with the minicomputer/stylus instrument system:

A. Allocation of the Interdata Model 3 memory during use by the step/roughness measurement software,

B. Flow diagrams of major programs or routines,

C. A memorandum by Philip G. Stein (NBS) which describes the properties and the use of an operating system he developed for the Interdata Model 3,

D. An annotated listing of all the step/roughness measurement software and of the operating system,

and

E. An instruction set listed by Op-code, for the Interdata model 3 with an excerpt on instruction word formates taken from the Interdata Model 3 manual.

The detailed information about the computer software which is given herein is presented so that the reader will have available an actual implementation of a total software system for computerized surface roughness measurement. Listings of the software in machine language should be of use to the reader whose minicomputer memory size is limited and thus is unable to support a compiler and to the reader considering the implementation of parts of the software system on a recent generation minicomputer. For those with assemblers and/or compilers, the flow diagrams and listings should enable the efficient writing of necessary programs. The incorporation of this appendix and motivated by the desire that one or all three of these routes be encouraged in every possible way. Thus, any questions pertaining to the software's operation, design or use would be welcomed by the author.

The documentation of computer programs in machine language is impossible without making reference to specific equipment and instruction sets by brand name. However, no judgement as to the quality or suitability of the equipment discussed here has been made by the National Bureau of Standards, and no recommendation, favorable or otherwise, should be implied by this report.

The software is neither as elegant nor as efficient in many places as I would have desired. This statement is not a criticism of the work contributed by others. It is more of an encouragement to the user to take the programs as they stand and to make as many improvements as his time allows. The software is however a system of programs which has passed every functional test conceived to check its

operation. As described in the body of the report, these checks have included static and dynamic checks such as: (1) If step data goes from an ordinate (a-c) to a second ordinate (b-c), is the calculated step height constant and equal to b-a for all values of c within the data-range of the computer? (2) If the slope on the "left" side of the step increases, does the step height decrease?; and (3) Does the AA values calculated from known waveforms correlate with analytically calculated values?

I wish to acknowledge with thanks the many contributions to this software system by NBS associates. Their advice and, in many cases, their contribution of complete program listings made the writing and revising of the system software a much easier task. Contributors to the software were Dennis A. Swyt (DAS), Philip G. Stein (PGS), Chris E. Kuyatt, (CEK), Nils Swanson (NS), Carol Young (CY), and E. Clayton Teague (ECT). The major source of each program or routine is acknowledged in the listings with the initials just given for each contributor. Final responsibility for the correctness of the listings and documentation should however fall upon me since many relocations, renamings and minor revisions have been made on the original programs.

Throughout this appendix the following abbreviations and notations are used:

R0—RF ; general registers 0 thru F,

MMY ; memory,

TTY ; teletype,

CRL ; carriage return and line feed,

SP ; space,

ADC ; analog to digital converter,

ASCII ; American Standard Code for Information Interchange,

ACF ; Autocorrelation Fuction,

ADF ; Amplitude Density Function,

LSQ ; Least squares, and

[] or () ; for use other than in mathematical equations should be read-contents of a register or a memory address, i.e., [(0560)+(RC)] should be read; contents of the memory address obtained by adding the contents of memory address 0560 to the contents of register C.

II. Allocation of Interdata 3 Core Memory During Use By The Software For Surface Roughness Measurement.

(0000-004F) = General Registers 0-F, Hardware Registers and Program Status Words

(0050-0078) = "50" Loader to input paper tapes at the TTY

(0080-02FE) = Main Step/Roughness Calibration Program

Input - H0, A1, A2, and Units at TTY,
Data and Interrupt from ADC,
Decisions indicated at TTY for roughness or steps

Output - Step heights, KCAL, and AA values at TTY

Storage - Step Data, (1000-1400),
Step heights, (0B60-0B82),
Profile data, (1000-2000),
AA values, (0B14-0B5E),
KCAL value, (0BAC-0BAE),
H0, (0B94),
A1, A2; (0B90-0B92),
Units; (0BB0),
Temporary storage of AA - (0B0E),
Indices for taking AA - (0B0A-0B0C),
Flag for Bypassing KCAL calculation -(0BB2),

(0300-034C) = Main Step Height Program

Input - A1 and A2 from (0B90-0B92),
Step Data at (1000-1400)

Output - Hexadecimal step heights and slopes and intercepts of least squares lines on each side of step

Storage - Hex step heights; (0B60-0B82)
Slopes and intercepts; (0B96-0BA4)

(0350-03FA) = Calculate Hex Step Heights Routine

Input - A1 and A2 from (0B90-0B92);
Slopes, a1 and a2, and intercepts,
b1 and b2, from (0B96-0BA4)

Output - Hex step heights stored at (0B60-0B82)

(0400-043C) = Calculate KCAL Routine

Input - Decimal H0 from (0B94),
HM from (0B80-0B82)

Output - KCAL stored at (0BAC-0BAE)

(0440-0492) = Convert Hex Steps to Decimal Using KCAL, Roundoff and Print Results

Input - Hex step heights from (0B60-0B82),
KCAL from (0BAC-0BAE)

Output - H1, H3, H5, H7, HM in decimal at TTY

(04A0-0512) = AA Calculation

Input - Profile data from (1000-3000)

Output - AA value of data in (R5)

(0518-0552) = Convert AA in Hexadecimal to Decimal and Print Result

Input - AA value from (R5)
KCAL from (0BAC-0BAE)

Output - Decimal value of AA at TTY

(0560-0634) = Least Square Fit to Data from (0590) to (0590) + (0578)

Input - Starting point of data, (0590);
Number of points, (0578).

Output - Slope of line fit x 1000 → (R45),
Intercept of line fit x 1000 → (R01)

(0640-0658) = Routine to Set Parameters for Reading Step Data from ADC into MMY

(065A-0672) = Routine to Set Parameters for Reading Roughness Data from ADC into MMY

(0680-0A80) = Storage Area For Amplitude Density Function or Auto-correlation Function Calculations

(0A80-0B00) = Presently Unused

(0B0A-0BB2) = Storage for Various Parameters of Main Step/Roughness Program

(0BB4-0BFE) = Presently Unused

(0C00-0CBC) = Program to Calculate and Plot the Autocorrelation Function of Data in MMY Locations (1000-3000)

Input - Data in (1000-3000),
Shift increment = (0C06)

Output - $(RMS)^2$ value of data adjusted by KCAL at TTY, 8 bit precision plot of calculated data at strip chart recorder.

Storage - Calculated data for 512 shifts stored at (0680-0A80)

(0D00-0D) = Routine to Calculate Mean and Standard Deviation of a Set of AA Values

Input - Number of AA Values upon query at the TTY
AA values from (0B14-0B5E)

Output - Mean AA value and Standard Deviation of Data set from Mean

(0E00-0EF0) = Calculate Derivative and Mean Wavelength of Data in (1000-3000)

Input - Profile Data in (1000-3000)

Output - Mean slope of Data,
Mean wavelength of Data based on KCAL and sample spacing for usual roughness speed
Both parameters are output at TTY

Storage - Data in (1000-3000) is destroyed

(0EF0-0EFE) - Presently Unused

(0F00-0F3C) = Routine to Store Contents of all Registers Except Register B

(0F48-0F84) = Routine to Restore Contents Stored by 0F00

(0F88-0FB4) = Register Storage For Registers 0, 1, 2, 7, 8, 9, A, C, D, E, and F.

(0FC2-0FEE) = Restore Contents Stored by 0F88

(1000-3000) = Data Storage for Step Input and for Roughness Profiles

(1500-1670) = Program for Measuring the Height of Three Consecutive Steps

Input - Step data from (1000-1400), Location of Step and period of steps from TTY, KCAL from (0BAC-0BAE)

Output - Step height values at TTY

Storage - Usual step height storage plus (2000-2006) for index and other parameter storage.

(2500-25B2) = Program for Taking Statistics on Step Profile Data

Input - Step location entered into (251A) and (2522) using monitor, step profile data from (1000-1400), and interrupt from ADC

Output - Slope and intercept of LSQ line fit to each side of step, KCAL, and HM; all in hexadecimal, to TTY.
Step height in decimal relative to first step input, is also printed at TTY

Storage - No storage beyond usual step height program requirements

(3010-3020) = Routine to Wait for an Interrupt, Then Read Data at the ADC

(3022-306C) = Routine to Load Data From ADC

Input - Digitized data from ADC based on an analog signal at the appropriate ADC channel number for duration of sampling time

Output - Data stored at 1000 to 1000 + (302E)

(306E-30B8) = Wait for an Interrupt at ADC, Mark time for 1 Second, Then Exit

Input - Interrupt at ADC

(30BA-30E4) = Routine to Read 4 Hex Characters at TTY into R2

Input - 4 characters just preceding a CR or SP at TTY

Output - Characters read are left in (R2)

(30E8-310C) = Routine to Read 2 Characters at TTY and store ASCII code at 0BB0

(3110-3120) = Print Single Space Routine

(3122-3146) = Print 2N+1 Spaces Routine with (3130) = N.

(314A-316A) = Print Two Characters from ASCII code in (R5).

(3170-318E) = Routine to Print "UNITS"

(3190-31AC) = Routine to Print "AA -------- UNITS" + CRL

(31B0-31D4) = Routine to Print "H OR R?"

(31D6-31F4) = Routine to Print "MORE?"

(31F8-3216) = Routine to Print "ENTER"

(3218-3236) = Routine to Print "DATA"

(3240-327F) = Routine to Print "H1---H3---H5---H7---HM"

(3280-32CA) = Routine to Print "STEP/ROUGHNESS CALIBRATION"

(32D0-32F0) = Routine to Print Hexadecimal Form of Contents of R2 and R3

(3300-33FE) = Routine to Convert Binary Number to Decimal Number

Input - (R45) of either sign, full 31 bits

Output - (R123) and at MMY addresses from [(336A)+(R8)] to [ ] + C

Storage - MMY indicated in output plus (3530-354C) for hexadecimal equivalents of powers of 10

(3408-34CE) = Routine to Multiply Unsigned (R5) x (R9) → (R89)

(34D4-3524) = Routine to Multiply (R34) x (R6.7) → (R23.45)

Input - Both (R34) and (R6.7) must be positive

(3530-354C) = Storage for Hexadecimal Equivalents of powers of 10

(3550-355C) = Storage of Binary to Decimal Conversion

(3560-358C) = Routine to Multiply (R234) by 2

(3590-35C0) = Multiply (R45) x (R6), Put Result in (R345)

Input - (R45) either sign, (R6) assumed positive

Output - Result → (R345); Flag = 8000 put in (35EE) if (R45) negative.

(35C4-35F4) = Routine to take Absolute Value of (R45) and Set Flag at 35EE if contents negative.

(35F8-3620) = Routine to Change Sign of (R45) if Flag at 35EE is Set

(3622-3642) = Routine to Convert Single Precision (R3) to Double Precision in (R23)

(3644-3696) = Routine to Convert Decimal (R4) to Hexadecimal at R1.

Input - Assumes (R4) are positive

(36A0-374C) = Routine to Divide 2 Hex Halfwords by 1 Hex Halfword.
(R45) ÷ (R3) → (R45)

Input - (R45) may have either sign, but (R3) are assumed positive. Only the 30 most significant bits of (R45) are used.

(3750-37E2) = Program for Printing and Punching contents of MMY Locations 1000-3000

Input - 12 bit left-justified binary numbers of either sign from (1000-3000)

Output - Four digit signed decimal numbers printed (and punched) in a format of number - space with ten number per line. A space is printed for a positive sign.

(3800-3888) = Program to Calculate the Amplitude Density Function For Data Stored at (1000-3000)

Input - Data at (1000-3000)

Output - Calculated data is stored at (0680-0A80), then plotted on strip chart recorder to an 8 bit precision.

(3900-3916) = Program to Clear (1000-3000)

(3918-3946) = Program to Initialize Plot Routine for Bipolar Numbers and to Plot Data in (1000-3000) on Strip Chart Recorder

(3950-39CF) = Plot Routine to Drive Polombo's Waveform Receiver. This Routine takes Halfwords from Specified Memory Locations Using the 8 Most Significant Bits (Including the Sign Bit) and Biases the Data to have a Range from 4 to 255.

Input - (3978) = Starting address of data to be plotted, (397C) = Number of shifts needed to obtain 8 bit significance for input, (3984) and (398C) = Maximum value to be plotted, (3990) = Bias, and (399E) = 2X number of points to be plotted. Note that the bias and maximum value should be adjusted so that; (bias) + (MAX) = 255.

Output - Minimum (4) and Maximum (255) values for approximately 8 seconds, followed by data formatted according to input just described.

(39D4-39FE) = Routine to Print "MEAN AA ="\

(3A00-3A2A) = Routine to Print "VARIANCE ="

(3A30-3A48) = Program to Print "ERROR," then return to monitor.

(3A50-3A70) = Print "MM" and Increase Power of 10 in Units by 1 if for AA

(3A80-3FCE) = HEXADECIMAL MONITOR
See annotation and Philip Stein's memorandum in the following pages for use and explanation.

III. Flow Diagrams of Major Programs or Routines

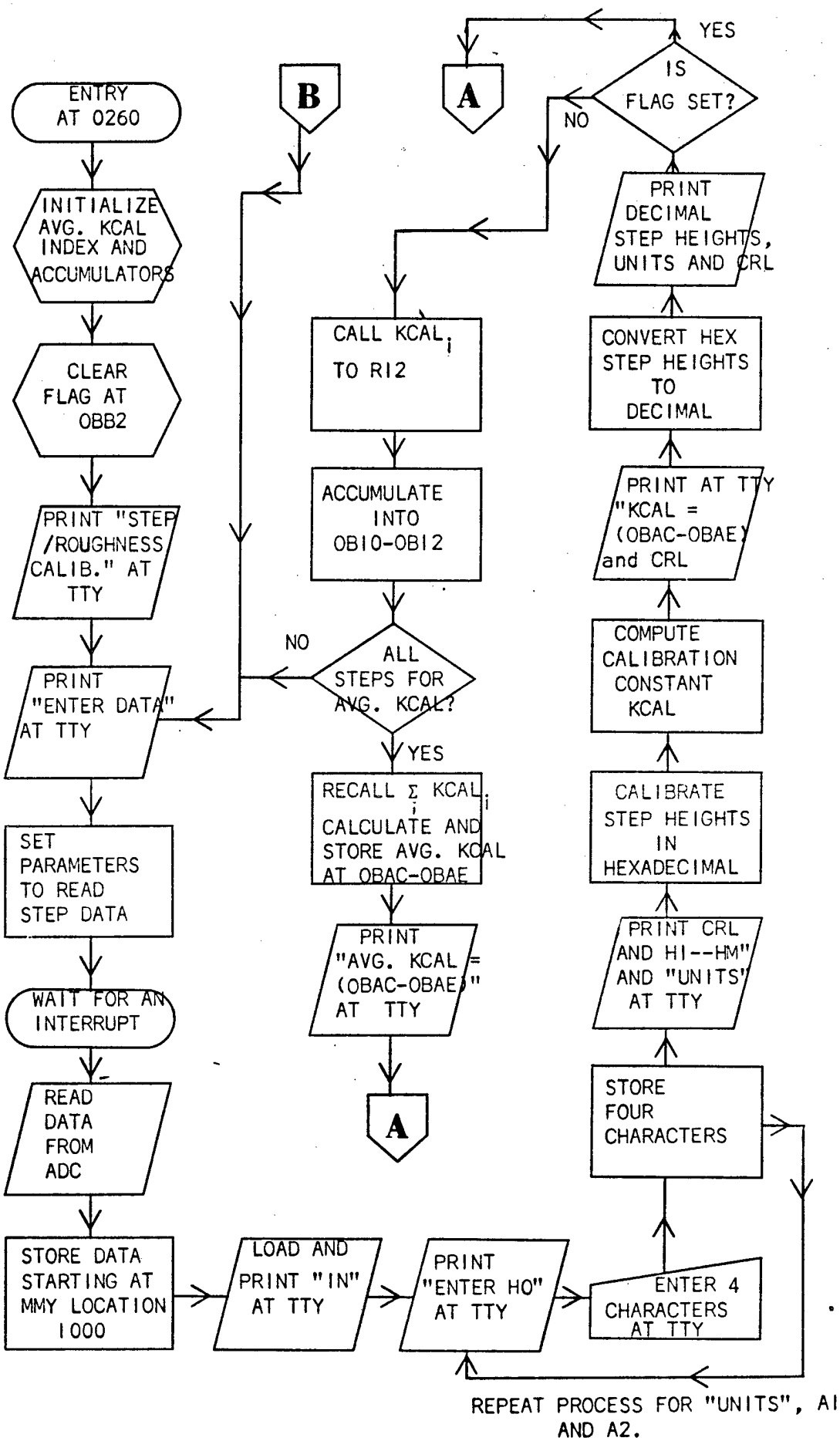


Figure 1: Main Step/Roughness Program, Page 1 of Flowchart.

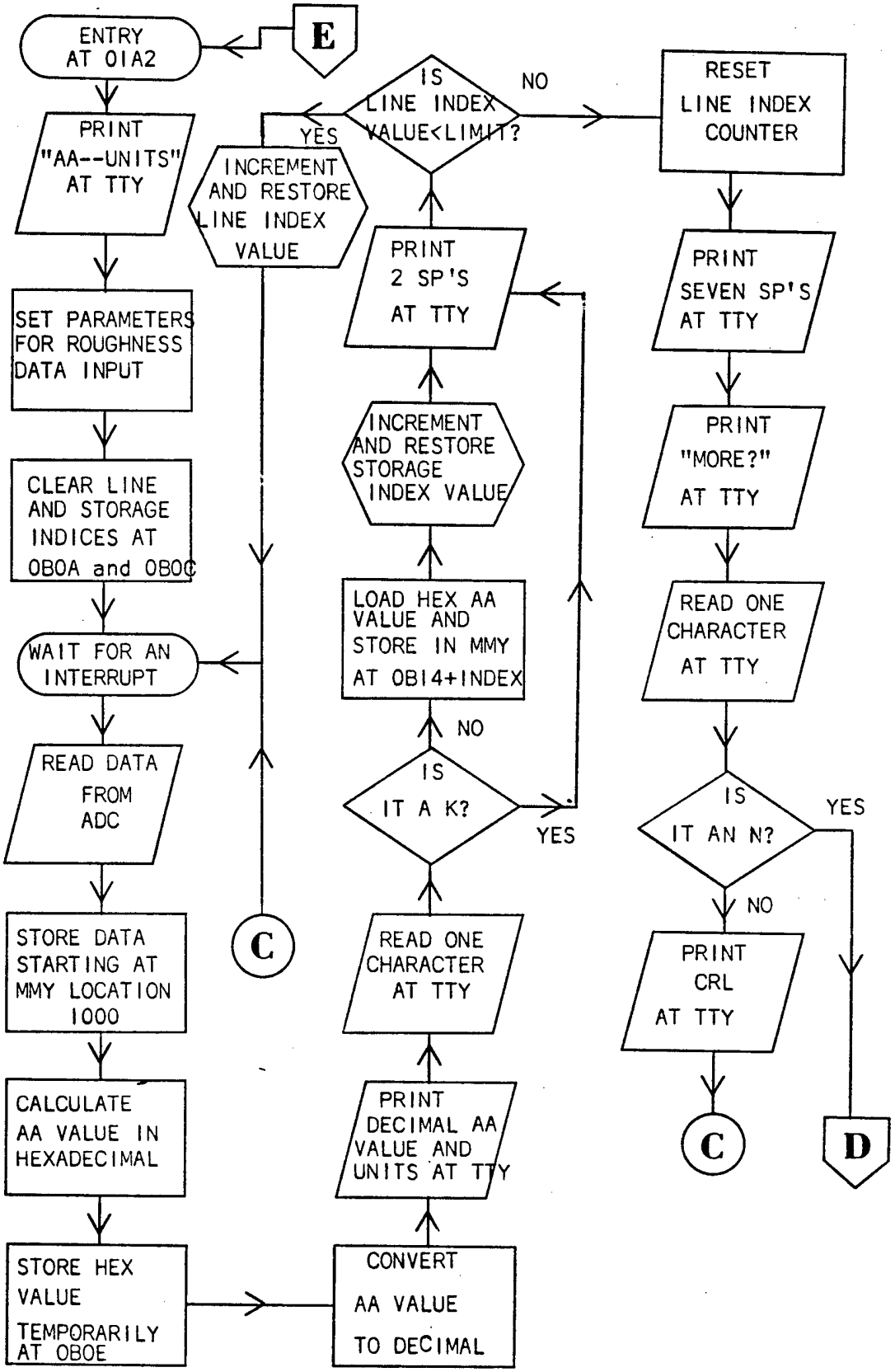


Figure 2 : Main Step/Roughness Program, Page 2 of Flowchart.

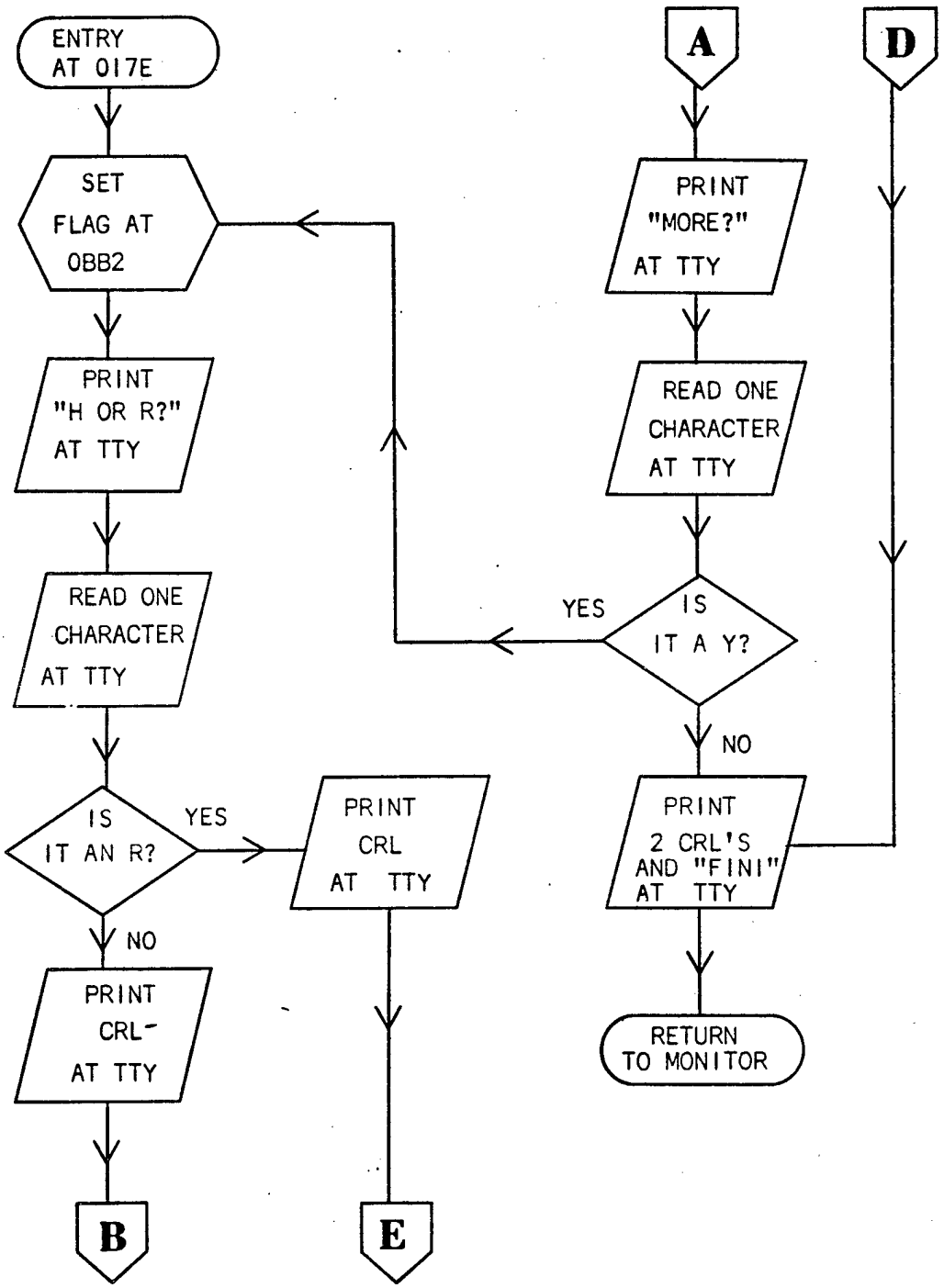


Figure 3. Main Step/Roughness Program, Page 3 of Flowchart.

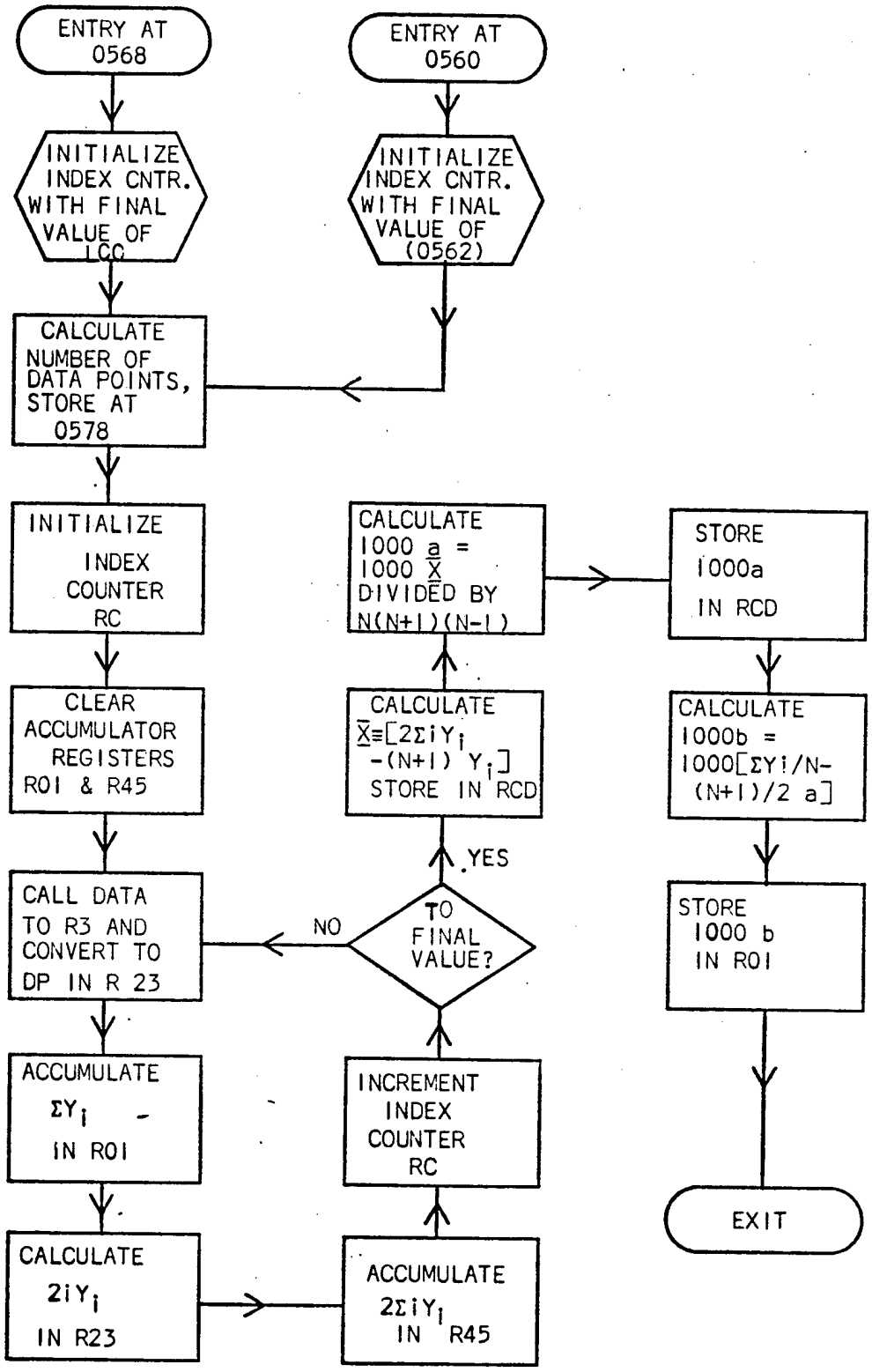


Figure 4: Flowchart of Least Squares Routine.

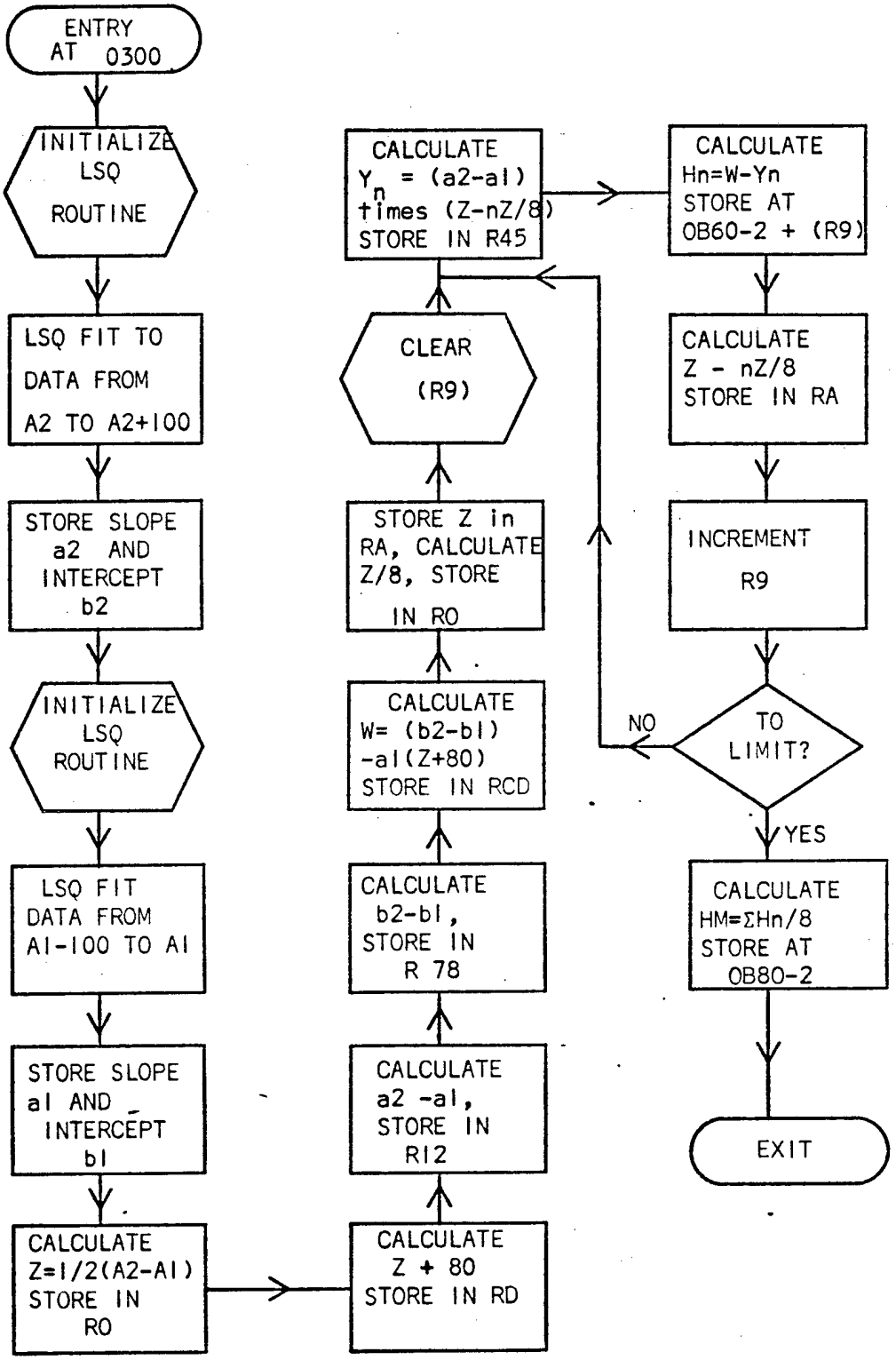


Figure 5: Flowchart of Routine to Calculate Step Heights.

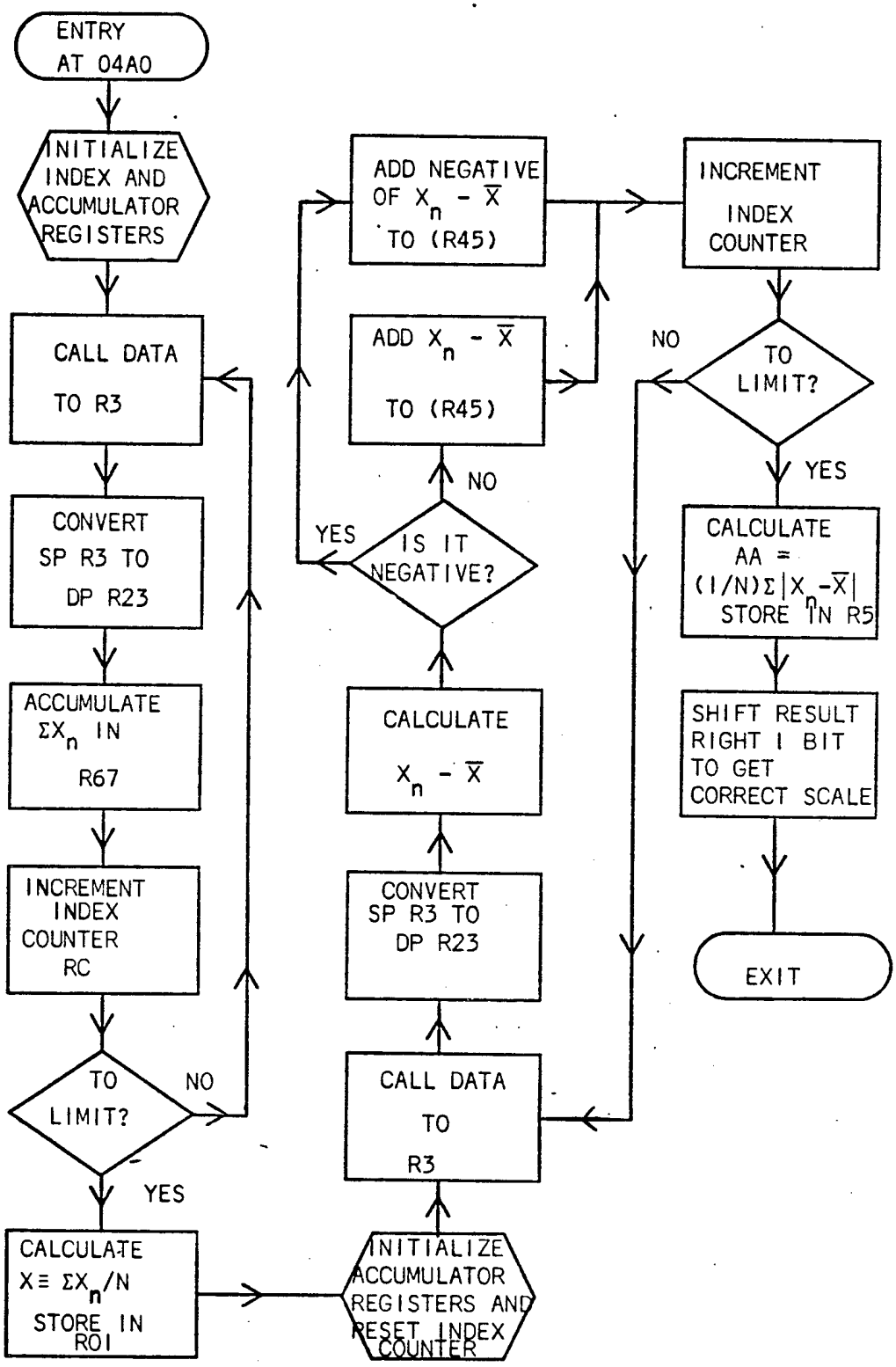


Figure 6 : Flowchart of Routine To Calculate AA Value.

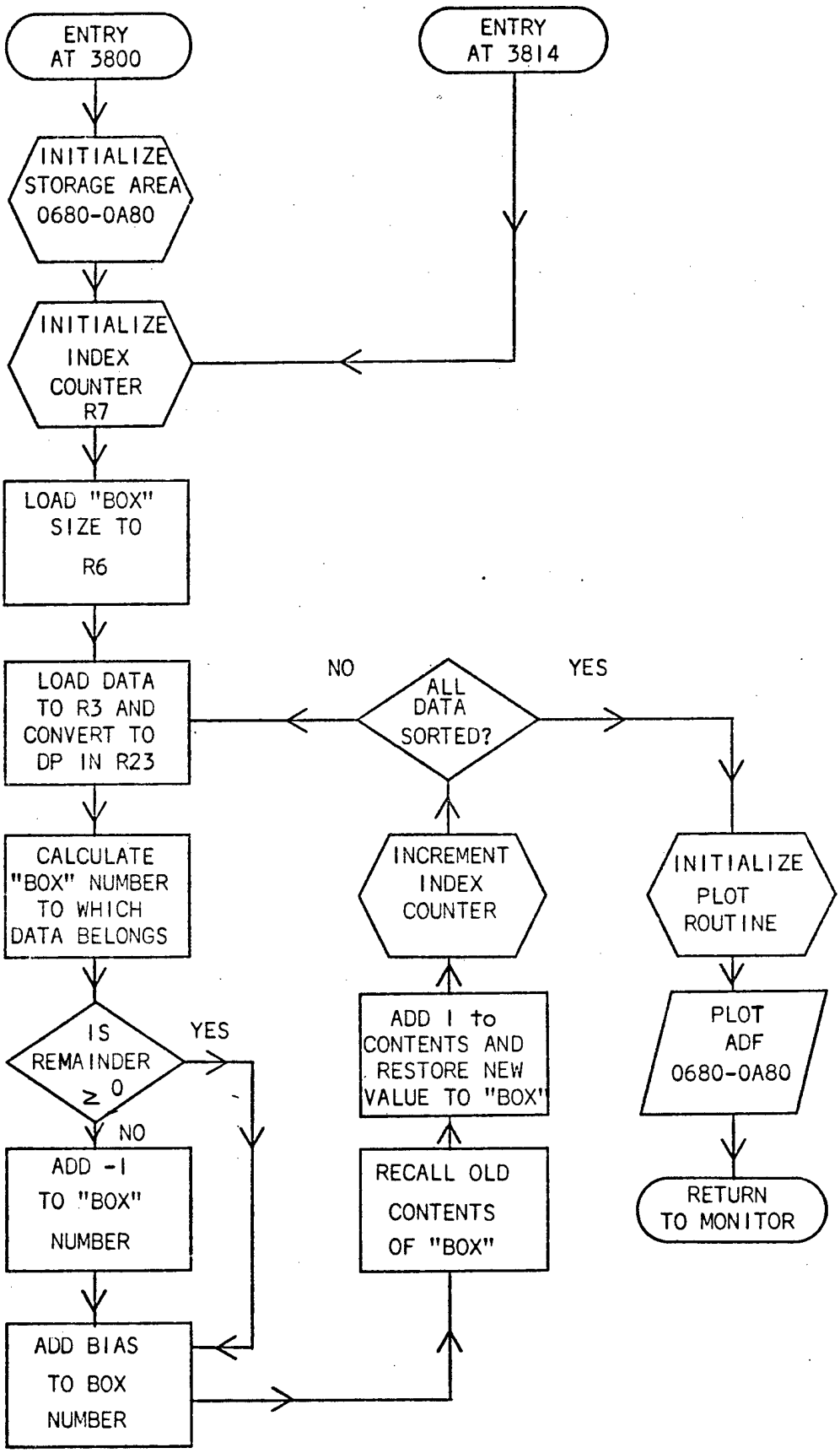


FIGURE 7: Flowchart of Amplitude Density Function Calculation.

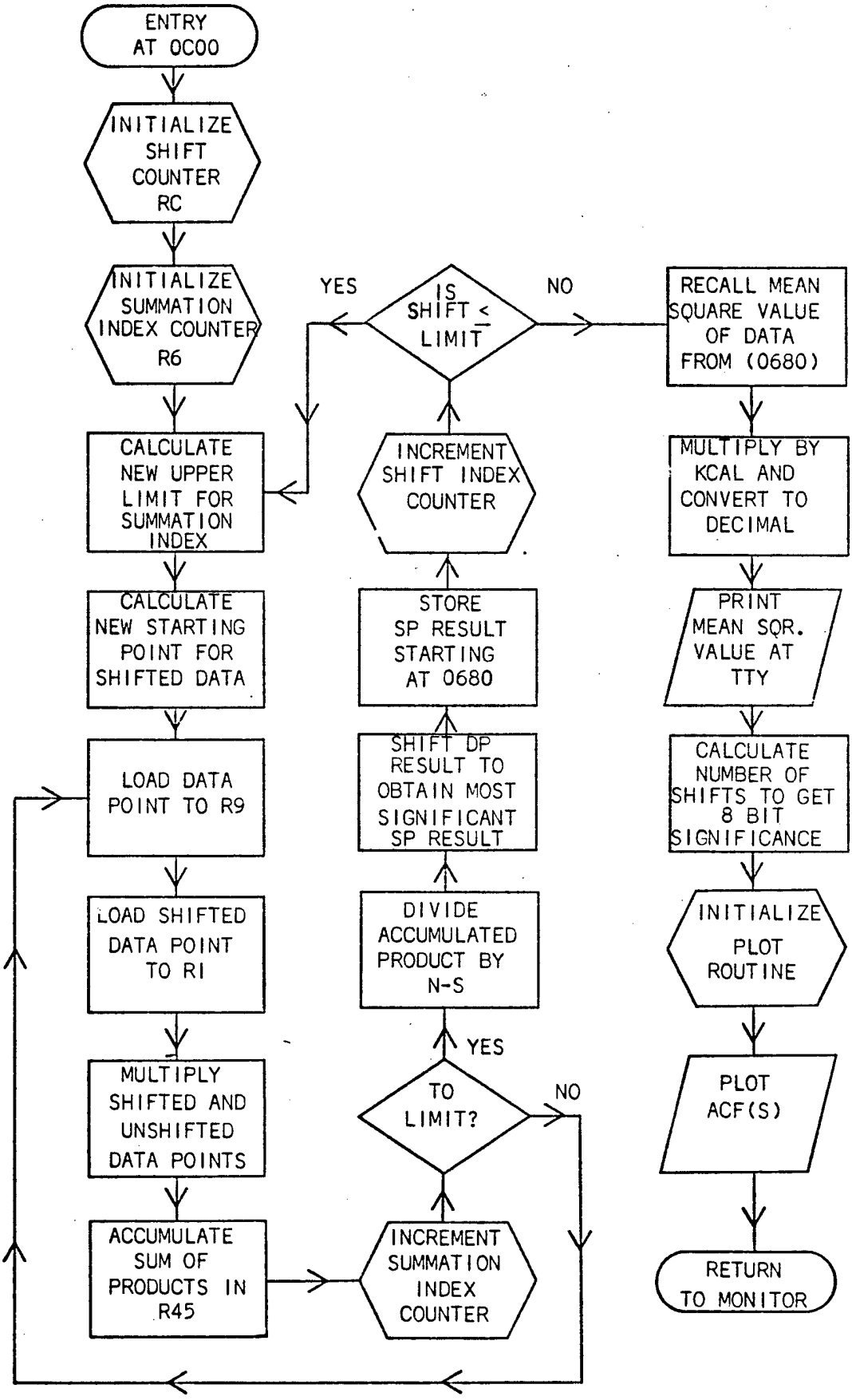


Figure 8: Flowchart of Autocorrelation Function Calculation

IV. Operating System

NATIONAL BUREAU OF STANDARDS
Center for Computer Sciences and Technology
Information Processing Technology Division
Washington, D. C.

February 4, 1969
650.1/1

TECHNICAL MEMORANDUM

From: Philip G. Stein (650.01)

Subject: A HEXADECIMAL MONITOR SYSTEM FOR THE INTERDATA COMPUTER

During 1968, a computer program was developed at the National Bureau of Standards to allow more convenient use of the INTERDATA model 3 computer. Programs may be prepared, documented, debugged, and run directly from the teletype keyboard. All data and instructions handled by the system are in absolute, hexadecimal format. No symbols are accepted or interpreted except those which are used directly as system commands.

This publication is designed to be used as an operators manual for the system, and includes sufficient documentation to allow a programmer to duplicate it, change it, or incorporate it into a larger structure applicable to special problems.

Although it is impossible to prevent referring to specific equipment by brand name when documenting a computer program designed for a given machine, no judgement as to the quality or suitability of the equipment discussed here has been made by the National Bureau of Standards, and no recommendation favorable or otherwise, should be implied by this report.

It is assumed that the reader is fully familiar with the operation of the INTERDATA model 3, and with the instructions presented in its reference manual, Interdata publication 29-004. According to the manufacturer, this program will run without modification on the model 2 and model 4 as well.

PART I - STORAGE ALLOCATION

All of the descriptions contained here refer to a system written for a computer with 16K bytes or core memory. The system may be used with any memory of 2K bytes or more, and may be relocated by hand to fit other memory sizes. Paper tapes for an 8K system and a 16K system may be obtained on application to the author.

Memory is reserved in a 16K machine as follows:

0000-004F	System Registers

0050-007F	'50' loader, as modified for use with this monitor
0080-397F	User program space
3A80-3E7F	Monitor system
3E80-3F6F	Subroutines-Used by monitor but may be called from anywhere
3F70-3F7F	Subroutines return push-down stack
3F80-3FCF	Bootstrap Loader area
3FD0-3FFF	Register SAVE area

In addition, the system uses all of the 16 registers when in operation. It is never necessary to clear or preset any registers when entering the system, but programs called by the system should never depend on the presence of previously stored data in the registers.

The system subroutines are general purpose utility programs which will be very convenient for most programmers to call from their own programs. They use some registers, and these are called out in the description of the specific routines.

A programmer wishing to write in such a manner as to be most compatible with the monitor should follow these register assignments:

B: Subroutine return register. Calls: 41B0 Returns: 030B

C: Initial address register for loops

D: Increment for loops

E: Final Address register for loops

F: Input/Output device number

Subroutines are nested by placing successive returns in a stack located between 3F70 and 3F7F. Programmers wishing to place returns in other registers should put them in 6 and 7 for minimum interference with the monitor.

PART II - LOADING

The monitor system contains its own bootstrap loading program for handling paper tape inputs. The loader is read into 3F80-3FCF by the resident Interdata '50' loader. Make the following changes in the '50' loader.

ADDRESS	WAS	SHOULD BE
0052	????	3F80
005A	????	3FD0
0076	????	3F80

Then follow the instructions for the '50' loader. After the full loader has been read, there is a blank space, followed by the program. When this area is encountered, front-panel display lights (DIS 2 bits 8:15) should begin to flash. When the read is finished, these lights will all be out if no read errors were made. Stop the tape reader after the last character has been read.

The above procedure is used not only for loading the monitor tape itself, but also for loading program tapes produced in binary form by the monitor.

PART III - USERS MANUAL

After loading, manually restart the computer at location 3C80. The computer will type

```
>
```

as it will every time it expects an input from the operator. The operator may then begin typing data, instruction, or monitor commands. All keys, printing or non-printing, on the teletype are active. Those that are not used for Hexadecimal numbers or for monitor commands will cause erroneous data to be entered. Accidental operation of these keys should not be ignored, but corrections should be made.

When typing data or instructions, the monitor only saves the last four characters typed. For this reason, if an error is made, simply continue typing until the last four characters typed are correct. The input data register is cleared after every monitor command, and data is brought into it right-justified. The user may therefore type without bothering with leading zeros. They will be included automatically. After four correct hexadecimal characters are typed into the data register, one of many system commands may be given.

SYSTEM COMMANDS: POINTER

R - The data register is loaded into a pointer address register. This pointer indicates where data or instructions will be stored or retrieved.

I - The address pointed to by the pointer is incremented by one halfword.

RUBOUT - The address pointed to by the pointer is decremented by one halfword, and the character ↑ is printed.

W - The current contents of the pointer are printed, and nothing is changed.

SYSTEM COMMANDS: INPUT

CARRIAGE RETURN - The data in the input register is stored at the location indicated by the pointer. Data previously located at that address is destroyed. The position of the pointer is incremented by one halfword, and the system returns a >.

SPACE - Has the same effect as carriage return, except that the system does not return any printed characters.

Note: Using the above, instructions may be typed exactly as they appear on the standard coding sheet. i.e.

```
C8F0 1234
4330 1000
033C
9BF8
41B0 2222
```

and data may be typed

```
1234 5678 9ABC DEF0 1234 5678 9ABC DEF0
1234 5678 9ABC DEF0 1234 5678 9ABC DEF0
```

with no change in the system.

SYSTEM COMMANDS: OUTPUT

P - Prints the contents of memory, beginning at the pointer and ending at the halfword preceding the data register. The routine is therefore called -

```
1000R
1020P
```

which will immediately begin printing in program format: i.e.:

```
1000 C8F0 1234
1004 41B0 2222
1008 9EFF
100A 0309
100C 4300 1C8A
```

Addresses are printed on every line. Halfword and fullword instructions

are printed so that each occupies a single line, and the address computations are adjusted automatically. Printing may be terminated prematurely by depressing one of the console buttons 8 through 15.

L - Prints the contents of memory, beginning at the pointer and ending at the halfword preceding the data register. The routine is therefore called

```
1000R
1020L
```

which will immediately begin printing in data format: i.e.:

```
1000  1234 5678 9ABC DEF0 1234 5678 9ABC DEF0
1010  1234 5678 9ABC DEF0 1234 5678 9ABC DEF0
```

Addresses are printed at the beginning of every line, and the format is fixed regardless of what is being printed. Printing may be terminated prematurely by depressing one of the console buttons 8 through 15.

K - Prepares a binary bootstrap tape and a manuscript tape which may be loaded with the same '50' loader that loaded the monitor itself. It is called in the same way as the P and L routines, but may not be interrupted from the console switches. Before typing K, turn on the tape punch by striking control R (ASR 35) or punch ON (ASR 33). On the ASR 35, after striking K, rotate the MODE switch to TTR.

SYSTEM COMMANDS: CONTROL

G - Causes the computer to jump to the address in the data register. If this address is greater than 3B00, the command is ignored. If the address is accepted, the computer will ring the teletype bell and wait for a console switch (8 through 15) to be pressed before executing the jump.

> - Causes the computer to reinitialize the monitor and restart it. When a new page or other circumstance makes it unknown whether the system is running or where it is, typing > will return

```
>
```

and restart the system. It is suggested that all other routines which and call the system be written so that typing > will jump to 3C80 and restart the monitor.

PART IV - LINKING

Users who which to add their own routines to be called by monitor commands should do so as follows:

There are four blank spaces in the command tables. Place the ASCII code for the letter command, and the address to which a jump is desired when that letter is typed, into memory as shown:

Letter Command in BYTE	JUMP ADDRESS in HALFWORD
3CD0	3CF4
3CD1	3CF6
3CD2	3CF8
3CD3	3CFA

Your routine may retrieve the pointer from register 5 and the input data buffer (four characters, converted to hexadecimal) from register 4.

To return to the monitor, reinitialize, and print

>

Jump to 3C8A. To return without the printing, jump to 3C96. No other addresses in the monitor snould be used as entry points.

PART V - UTILITY SUBROUTINES

These programs should be useable by every program for handling input/output and data conversion chores. They are listed below, and include calls, location of input and output argument, and lists of registers destroyed by their use.

Name	INPUT ARGUMENT (register or other)	OUTPUT ARGUMENT (register or other)	DESTROYS register
PRINT 41B0 3E80	0 (8-15)	teletype	0, 8, F
READ 41B0 3E8A	teletype	0 (8-15)	0, 8, F
ASCII-HEX 41B0 3EA4	0 (8-15)	1 (12-15)	0, 1
HEX-ASCII 41B0 3EC8	0 (12-15)	1 (8-15)	0, 1
LOOP 41B0 3EE2	Panel Buttons 8-15	Condition Code	0

NAME	INPUT ARGUMENT (register or other)	OUTPUT ARGUMENT (register or other)	DESTROYS register
PRINT4 41B0 3EF0	2	teletype	0, 1, 2, 3, 8, F
LEADER 41B0 3F26	0 (8-15)	teletype	0, C, D, E, F
CARL 41B0 3F44		teletype	0, 8, F

PRINT: Prints the contents of register 0 (8-15, in ASCII) on the teletype.

READ: Waits for a character to be typed on the teletype, then loads it into 0 (8-15).

ASCII-HEX: Interprets the ASCII characters 0-9 and A-F and converts them to Hex. This routine does not do correct processing on any other character.

HEX-ASCII: Converts the sixteen hex characters to their ASCII equivalents, With the parity bit always set.

LOOP: Senses the front panel buttons, and sets the condition code equal to 0 if none of them are pushed.

Print4: Prints the entire halfword located in 2 on the teletype as four Hex characters.

LEADER: Punches six inches of tape containing the character located in 0 (8-15).

CARL: Causes the teletype to carriage-return, line feed.

PART IV - DOCUMENTATION

An annotated listing of the monitor program is included with the complete listing of the software in this appendix. They should be self-explanatory to a programmer familiar with the Interdata 3. Any questions concerning the monitor, its operation, design, or use would be welcomed by the author. Details of the tape punch and print formats may be found by following the process by which they are generated, as seen in the listings.

V. Brief Description of Op-Codes and Instruction Word Formats

These excerpts from the Interdata Model 3 Manual are given to make the listings independently readable. Further details about the instruction set, instruction word format and the model 3 minicomputer may be obtained from Publication Number 29-004R02, published by Interdata Inc., 2 Crescent Place, Oceanport, New Jersey 07757I, phone number (201) 229-4040.

INSTRUCTION WORD FORMATS

Instructions in INTERDATA Systems have three formats:

1. Register to Register [RR]

2. Register to Indexed Memory [RX]

3. Register to Storage [RS]

In general, each format specifies three things: The operation to be performed, the address of the first operand, and the address of the second operand. The first operand is normally a General Register which contains the result of a previous operation. The second operand is normally the contents of a General Register, the contents of a core memory location, or a data constant used as the other participating operand.

A 16-bit halfword format is used for register to register operations. A 32-bit fullword format is used for the register to indexed memory, and the register to storage formats. The specific formats are shown in Figure 9.

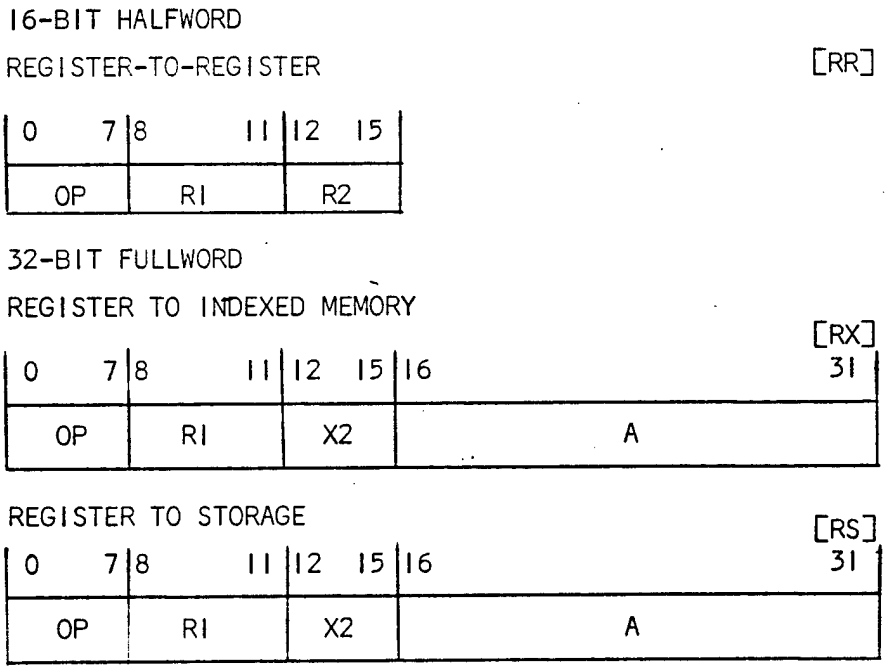


Figure 9: Instruction Word Formats

The 8-bit OP field in all three formats specifies the machine operation to be performed. The operation code can be written as two hexadecimal characters.

The 4-bit R1 field in the three instruction formats specifies the address of the first operand. The R1 field is normally the address of a General Register and is written as one hexadecimal character.

The 4-bit R2 field in the RR instruction format specifies the address of the second operand. The R2 field is always a register address and is written as one hexadecimal character.

The 4-bit X2 field in the RX and RS formats specifies a General Register whose content is used as an index value. The X2 field is always the address of a General Register and is written as a single hex character.

The 16-bit A field specifies a memory address in the RX format, or contains an integer value to be used as an immediate operand in the RS format. It is written as a string of four hex characters.

The RR instructions are used for operations between two registers. The first operand is the contents of the register specified by the R1 field of the instruction word. The second operand is the contents of the register specified by the R2 field.

The RX instructions are used for operations between a register and memory with the option of indexing. The first operand is the register specified by the R1 field of the instruction word. The second operand is the contents of the memory location specified by the A field of the instruction word, or by the sum of the A field and the contents of the General Register specified by the X2 field if indexing is specified.

In the RS instruction, the first operand is the contents of the General Register specified by the R1 field of the instruction word. The second operand is the number contained in the A field, or the number generated by adding the A field to the contents of the General Register specified by the X2 field if indexing is specified. The second operand of an RS instruction specifies the number of bit positions in shift instructions, or forms the second operand in immediate instructions. An immediate operand is two bytes of data used as an operand and carried in the halfword address field itself. The value in the address field is treated as a signed integer instead of a memory location address.

For the Branch on Condition instructions the first operand is the M1 field. This field is a 4-bit mask which is to be tested against the condition code contained in the Program Status Word.

LIST OF INSTRUCTION SET FOR
INTERDATA MODEL 3

OP CODE	TYPE	INSTRUCTION
01	RR	Branch and Link
02	RR	Branch on True Condition
03	RR	Branch on False Condition
04	RR	AND Halfword
05	RR	Compare Halfword
06	RR	OR Halfword
07	RR	Exclusive OR Halfword
08	RR	Load Halfword
0A	RR	Add Halfword
0B	RR	Subtract Halfword
0C	RR	Multiply Halfword
0D	RR	Divide Halfword
0E	RR	Add with Carry Halfword
0F	RR	Subtract with Carry Halfword
40	RX	Store Halfword
41	RX	Branch and Link
42	RX	Branch on True Condition
43	RX	Branch on False Condition
44	RX	AND Halfword
45	RX	Compare Logical Halfword
46	RX	OR Halfword
47	RX	Exclusive OR Halfword
48	RX	Load Halfword
4A	RX	Add Halfword
4B	RX	Subtract Halfword
4C	RX	Multiply Halfword
4D	RX	Divide Halfword
4E	RX	Add with Carry Halfword
4F	RX	Subtract with Carry Halfword
90	RR	Unchain
92	RR	Store Byte
93	RR	Load Byte
96	RR	Write Block
97	RR	Read Block
9A	RR	Write Data
9B	RR	Read Data
9D	RR	Sense Status
9E	RR	Output Command
9F	RR	Acknowledge Interrupt
C0	RS	Branch on Index High

OP CODE	TYPE	INSTRUCTION
C1	RS	Branch on Index Low or Equal
C2	RX	Load Program Status Word
C4	RS	AND Halfword Immediate
C5	RS	Compare Logical Halfword Immediate
C6	RS	OR Halfword Immediate
C7	RS	Exclusive OR Halfword Immediate
C8	RS	Load Halfword Immediate
CA	RS	Add Halfword Immediate
CB	RS	Subtract Halfword Immediate
CC	RS	Shift Right Logical
CD	RS	Shift Left Logical
CE	RS	Shift Right Arithmetic
CF	RS	Shift Left Arithmetic
D0	RX	Store Multiple
D1	RX	Load Multiple
D2	RX	Store Byte
D3	RX	Load Byte
D5	RX	Autoload
D6	RX	Write Block
D7	RX	Read Block
DA	RX	Write Data
DB	RX	Read Data
DD	RX	Sense Status
DE	RX	Output Command
DF	RX	Acknowledge Interrupt

IV. Annotated Listings of Complete System Software

```
   0080R
>0640P        (0080-02FE) = MAIN PROGRAM
0080 0700    [DAS & ECT]   Clear flag
0082 4000 0BB2
0086 41B0 32A4             Print "STEP/ROUGHNESS CALIBRATION"
008A 41B0 3F44             Two carriage returns and line feed (CRL)
008E 41B0 3F44
0092 41B0 31F8             Print "ENTER"
0096 41B0 3218             Print "DATA"
009A 41B0 0640             Set parameters to read step
009E 41B0 3010             Wait for an interrupt and read step data
00A2 4200 0000             continue
00A6 C850 C9CE             Load "IN"
00AA 41B0 314A             Print
00AE 41B0 3110             Print space (SP)
00B2 41B0 31F8             Print "ENTER"
00B6 C850 C8CF             Load "HO"
00BA 41B0 314A             Print
00BE 41B0 3110             Print SP
00C2 41B0 30BA             Read 4 characters at teletype (TTY), step height
00C6 4020 0B94             Store at 0B94
00CA 41B0 31F8             Print "ENTER"
00CE 41B0 3170             Print "UNITS"
00D2 41B0 30E8             Read 2 characters at TTY (input units)
00D6 41B0 3110             Print SP
00DA 41B0 31F8             Print "ENTER"
00DE C850 C1B1             Load A1
00E2 41B0 314A             Print
00E6 41B0 3110             Print SP
00EA 41B0 30BA             Read 4 characters (input A1)
00EE 4020 0B90             Store at 0B90
00F2 41B0 31F8             Print "ENTER"
00F6 C850 C1B2             Load A2
00FA 41B0 314A             Print
00FE 41B0 3110             Print SP
0102 41B0 30BA             Read 4 characters (input A2)
0106 4020 0B92             Store at 0B92
010A 41B0 3F44             CRL
010E 4200 0000             Continue
0112 41B0 3240             Print "H1----H3----H5----H7----HM"
0116 41B0 3170             Print "UNITS"
011A 41B0 3122             Print 7 spaces
011E 41B0 0300             Calculate step heights in hexadecimal
0122 4200 0000             Continue
0126 4200 0000             Continue
012A 4200 0000
012E 41B0 0400             Compute calibration constant (KCAL)
0132 C850 CBC3             Load KC
0136 41B0 314A             Print
013A C850 C1CC             Load AL
013E 41B0 314A             Print
0142 C850 00BD             Load =
0146 41B0 314A             Print
014A 41B0 3110             Print SP
014E 4820 0BAC
0152 41B0 3EF0
0156 4820 0BAE             Print contents of 0BAC and 0BAE = (0BAC-0BAE)
015A 41B0 3EF0
015E C800 0000
0162 4000 0BB2             Flag Load - not presently used
```

```
0166 41B0 3F44          CRL
016A 41B0 0440          Convert hexadecimal steps to decimal and print
016E 4850 0BB0          Load units (those input previously)
0172 41B0 3A58          Print
0176 41B0 3F44          CRL
017A 4300 02F4          Branch to flag test
017E C800 1111          Load flag
0182 4000 0BB2          Store flag
0186 41B0 31B0          Print "H OR R?"
018A 41B0 3E8A          Read one character at TTY
018E C500 00D2          Compare it with an R
0192 4330 019E          Branch to AA routine if equal
0196 41B0 3F44          CRL
019A 4300 0092          Branch to start of step height routine
019E 41B0 3F44          CRL
01A2 41B0 3190          Print "AA------UNITS"
01A6 41B0 065A          Set parameters for roughness data input
01AA 07AA
01AC 40A0 0B0C          Clear accumulator and summation index
01B0 40A0 0B0A
01B4 41B0 3010          Wait for an interrupt and read data
01B8 41B0 04A0          Calculate AA value in hexadecimal
01BC 4050 0B0E          Store hex value temporarily at 0B0E
01C0 41B0 0518          Convert AA to decimal and print
01C4 41B0 3110          Print SP
01C8 4850 0BB0          Load units (those input previously)
01CC 41B0 3A50          Print
01D0 41B0 3E8A          Read one character at TTY
01D4 C500 004B          Compare with  a  K
01D8 4330 01F0          Branch around data storage if equal
01DC 4800 0B0E          Load hex value of AA to register 0 (R0)
01E0 4840 0B0A          Load storage index value to R4
01E4 4004 0B14          Store hex value at 0B14 + (R4)
01E8 CA40 0002          Increment index register by 2
01EC 4040 0B0A          Save index value at 0B0A
01F0 41B0 3110          Print SP
01F4 41B0 3110          Print SP
01F8 4840 0B0C          Load line index number into R4
01FC CA40 0001          Increment line index number by 1
0200 4040 0B0C          Save line index number at 0B0C
0204 C540 0003          Compare with number per line
0208 4280 01B4          If less than maximum take next entry
020C 0744               Clear R4 and
020E 4040 0B0C                Reinitialize line index number
0212 41B0 3122          Print  seven spaces
0216 41B0 31D6          Print "MORE"
021A 41B0 3E8A          Read one character at TTY
021E C500 004E          Compare it with an N
0222 4330 0240          Branch to print FINI if equal
0226 41B0 3F44          CRL
022A 4300 01B4          Return to taking next line of data
022E 0232               Not used
0230 41B0 31D6          Print "MORE"
0234 41B0 3E8A          Read one character at TTY
0238 C500 0059          Compare it with a Y
023C 4330 017E          Branch to "H OR R?" if equal
0240 41B0 3F44          CRL
0244 41B0 3F44          CRL
```

```
0248 C850 C6C9          Load FI
024C 41B0 314A          Print
0250 C850 CEC9          Load NI
0254 41B0 314A          Print
0258 4300 3C80          Return to monitor
025C 41B0 314A          NOT USED
0260 0700
0262 4000 0B10          Initialize index and accumulator registers
0266 4000 0B12
026A 4000 0B14
026E 4300 0080          Branch to start of step height routine
0272 4810 0BAC          Call KCAL_i to (R12)
0276 4820 0BAE
027A 4A20 0B12          Double precision accumulate into (0B10-0B12)
027E 4E10 0B10
0282 4010 0B10
0286 4020 0B12
028A 4810 0B14          Recall increment, compare and save index counter
028E CA10 0001
0292 4010 0B14
0296 C510 0005
029A 4280 0092          If i<5 repeat process of calibrating
029E 4840 0B10          Recall Σ KCAL_i, calculate average KCAL,
02A2 4850 0B12                  i
02A6 C830 0005
02AA 41B0 36A0
02AE 4040 0BAC          Store average KCAL at 0BAC-0BAE
02B2 4050 0BAE
02B6 C8A0 314A          Put print subroutine address in RA
02BA C850 4156          Load AV
02BE 01BA               Print
02C0 C850 4720          Load G space
02C4 01BA               Print
02C6 C850 CBC3          Load KC
02CA 01BA               Print
02CC C850 C1CC          Load AL
02D0 01BA               Print
02D2 C850 203D          Load space =
02D6 01BA               Print
02D8 4820 0BAC          Load (0BAC)
02DC 41B0 3EF0          Print
02E0 4820 0BAE          Load (0BAE)
02E4 41B0 3EF0          Print
02E8 41B0 3F44          CRL
02EC 41B0 3F44          CRL
02F0 4300 0230          Branch to "MORE?"
02F4 4800 0BB2          Recall Flag
02F8 4230 0230          If set branch to "More?"
02FC 4300 0272          If not set branch to AVG KCAL calculation
0300 40B0 034A  (0300-034C) = MAIN STEP HEIGHT PROGRAM
0304 4800 0B92  [ECT]   Recall A2
0308 4000 0590          Store at starting address of LSQ program
030C 41B0 0568          LSQ fit to data from A2 to A2 +100_x
0310 40C0 0B9E
0314 40D0 0BA0          Store slope a2 at 0B9E-0BA0
0318 4000 0BA2
031C 4010 0BA4          Store intercept b2 at 0BA2-0BA4
0320 4800 0B90          Recall AI
```

```
0324 CB00 0100          AI - 100x = New starting address of LSQ program
0328 4000 0590
032C 41B0 0568          LSQ fit to data from AI - 100x to AI
0330 40C0 0B96          Store slope al at 0B96-0B98
0334 40D0 0B98
0338 4000 0B9A          Store intercept bl at 0B9A-0B9C
033C 4010 0B9C
0340 41B0 0350          Calculate step heights in hexadecimal
0344 C8F0 0002          Restore TTY device number to RF
0348 C8B0 0122          Restore RB and return to main program
034C 030B
034E 0806    (0350-03FA) = CALCULATE HEX STEP HEIGHTS ROUTINE
0350 40B0 03F8 [ECT]
0354 4800 0B92          Recall A2 to (R0)
0358 4B00 0B90          A2 - AI → (R0) ≡ 2Z
035C CC00 0001          (R0)/2 = Z put into R0
0360 08D0               Save in RD
0362 CAD0 0080          Z + 80x put into RD
0366 4810 0B9E
036A 4820 0BA0          Slope a2 → (RI2)
036E 4B20 0B98
0372 4F10 0B96          a2 - al → (RI2)
0376 4870 0BA2
037A 4880 0BA4          Recall intercept b2 to (R78)
037E 4B80 0B9C
0382 4F70 0B9A          b2 - bl → (R78)
0386 4840 0B96
038A 4850 0B98          Recall slope al to (R45)
038E 086D               Z + 80x → (R6)
0390 41B0 3590          al (Z + 80x) → (R45)
0394 0B85
0396 0F74               (b2 - bl) - al(Z + 80x) → (R78)
0398 0799
039A 08A0               Save Z in RA
039C 08C7               Save (b2 - bl) - al(Z + 80x) in (RCD)
039E 08D8
03A0 CC00 0003          Z/8 →(R0)
03A4 087C               Recall (b2 - bl) - al(Z + 80x) to (R78)
03A6 088D
03A8 0841
03AA 0852               Recall (a2 - al) to (R45)
03AC 086A               Z - nZ/8 → (R6)      n = 0,1,---8
03AE 41B0 3590          (a2 - al) (Z - nZ/8) → (R45)
03B2 0B85
03B4 0F74               (b2 - bl) - al(Z + 80x) - (a2 - al) (Z -nZ/8) → (R78)
03B6 0BA0               (Z - nZ/8) → (RA)
03B8 4079 0B60
03BC 4089 0B62          Store result at 0B60-2 + (R9)
03C0 CA90 0004
03C4 C590 0020          Increment, compare (R9) to limit
03C8 4280 03A4          If (R9) are less than limit, return to calibrate new value
03CC 0744
03CE 0755
03D0 0766               Initialize accumulator and index R's
03D2 4A56 0B62
03D6 4E46 0B60          Double precision accumulate Hi's into (R45).
03DA CA60 0004             ΣHi → (R45)
03DE C560 0020
```

```
03E2 4280 03D2          ΣH_i → (R45) continued
03E6 C830 0008
03EA 41B0 36A0          ΣH_i/8 → (R45)
03EE 4040 0B80
03F2 4050 0B82          Store HM at 0B80-0B82
03F6 C8B0 0344          Restore RB and return to main program
03FA 030B
03FC 48B0 03C6 (0400-043C) = CALCULATE KCAL ROUTINE
0400 40B0 043A  [DAS]
0404 4840 0B94          Recall decimal HO to (R4)
0408 41B0 3648          Convert to hex value at (RI)
040C 0841
040E CF40 0000          Put hex value of HO in (R45)
0412 0755
0414 4860 0B80          Recall HM to R67
0418 4870 0B82
041C CF60 0003
0420 C470 E000          Multiple HM by 8 and put result in R3.
0424 CC70 000D
0428 0A67
042A 0836
042C 41B0 36A0          HO/8HM → (R45)
0430 4040 0BAC
0434 4050 0BAE          Store KCAL at 0BAC-0BAE
0438 C8B0 0132
043C 030B               Return
043E 48B0 40B0 (0440-0492) = CONVERT HEX STEPS TO DECIMAL
0442 0490       [DAS]   USING KCAL, ROUNDOFF AND PRINT RESULT
0444 C8C0 0000
0448 C8D0 0008          Initialize index counter
044C C8E0 0020
0450 4860 0BAC          Recall KCAL to (R67)
0454 4870 0BAE
0458 0722
045A 483C 0B60          Recall H_i to (R34)
045E 484C 0B62
0462 41B0 34D4          KCAL x H_i → (R345)
0466 C860 0008
046A 0777               Multiply result by 8 to account for 3 bit shift made in
046C 41B0 34D4                  KCAL calculation
0470 0788
0472 CD40 0001
0476 0E38               Round off R234 to R23 then move result to R45 [ECT]
0478 0E28
047A 0853
047C 0842
047E 41B0 3300          Convert hexadecimal in R45 to decimal in RI23 then
0482 41B0 32D0                  print result
0486 41B0 3110
048A C1C0 0450          Repeat for next H_i
048E C8B0 016E
0492 030B               Return to main program
0494 40C0 0452
0498 40D0 0454
049C 40E0 0456
04A0 40B0 0510 (04A0-0512) = AA CALCULATION
04A4 4200 1000  [DAS]
04A8 C8C0 0002          Initialize index counter
```

```
04AC C8D0 0002          Initialize increment value
04B0 C8E0 2000          Initialize final value
04B4 0766               Initialize accumulation registers
04B6 0777
04B8 483C 1000          Call data (1000 + (RC)) to (R3)
04BC CE30 0000          Null operation
04C0 41B0 3624          Single precision (R3) → double precision (R23)
04C4 0A73
04C6 0E62               ΣXn → (R67)
04C8 C1C0 04B8
04CC 4D60 04A6          X̄ = (ΣXn)/N → (R7)
04D0 0837
04D2 41B0 3624          Convert result to double precision
04D6 0802
04D8 0813               Save X̄ in (R01)
04DA 0744               Initialize accumulation registers
04DC 0755
04DE C8C0 0002          Reset index counter
04E2 483C 1000          Call data to (R3)
04E6 CE30 0000
04EA 41B0 3624          Single precision (R3) → double precision (R23)
04EE 0B31
04F0 0F20               Xn - X̄ → (R23)
04F2 4210 04FE          Branch if minus
04F6 0A53
04F8 0E42               Accumulate in (R45)
04FA 4300 0502          Branch to return
04FE 0B53
0500 0F42               Add negative of Xn - X̄ to accumulator
0502 C1C0 04E2          Return for next data entry
0506 4D40 04A6          (Σ|Xn - X̄|)/N → (R5)
050A CE50 0001          Data entry is 12 bit left justified, thus 1 bit right shift
050E C8B0 01BC                  gives same scale as KCAL
0512 030B               Return to main program
0514 0841
0516 0B43
0518 40B0 0550 (0518-0552) = CONVERT AA IN HEX TO DECIMAL AND PRINT RESULT
051C 0722       [DAS]
051E 0835
0520 0744
0522 C860 000A          Increase decimal precision of AA result by 1 position
0526 0777                       (051C-052A)
0528 41B0 34D4                  [Overflows if AA > 3/8 full scale]
052C 4860 0BAC          Recall KCAL to (R67)
0530 4870 0BAE
0534 41B0 34D4          KCAL x AA(hex) → (R234)
0538 0788
053A CD40 0001
053E 0E38               Round off to (R23) and put in (R45) [ECT]
0540 0E28
0542 0853
0544 0842
0546 41B0 3300          Convert AA(hex) to decimal
054A 41B0 32D0          Print result at TTY
054E C8B0 01C4
0552 030B               Return to main program
0554 0000
0556 48B0 0506
```

```
055A 030B
055C 0000 (0560-0634) =     LEAST SQUARE FIT TO DATA FROM (0590) to (0590) + (0578)
055E 091C      [ECT]
0560 C8E0 006C              0560 - Entry for any number of points specified by (0562)
0564 4300 056C
0568 C8E0 0100              0568 - Entry for usual 100x addresses
056C 080E
056E CE00 0001
0572 4000 0578              Calculate number of points and store at (0578)
0576 4200 0080
057A 40B0 0632
057E C8D0 0002              Initialize index counters (RE already set)
0582 C8C0 0002
0586 0700
0588 0711                   Clear accumulation registers
058A 0744
058C 0755
058E 483C 10A0              Recall data from [(0590) + (RC)] to (R3)
0592 41B0 3624              Single precision R3 → double precision R23
0596 0A13
0598 0E02                   Double precision accumulate into R01
059A 0C2C                   iY_i → R23
059C 0A53
059E 0E42                   Double precision accumulate into R45
05A0 C1C0 058E              ΣY_i → R01    2ΣiY_i → R45
05A4 08C4
05A6 08D5                   Save 2ΣiY_i in RCD
05A8 0840
05AA 0851                   Save ΣY_i in R45
05AC 4860 0578
05B0 CA60 0001
05B4 41B0 3590              (N + 1) ΣY_i → R45
05B8 0BD5
05BA 0FC4                   2ΣiY_i - (N + 1) ΣY_i ≡ X̄ → RCD
05BC 084C
05BE 085D
05C0 4830 0578              X̄/N → R45
05C4 41B0 36A0
05C8 C860 0060
05CC 41B0 3590              60X̄/N → R45
05D0 4830 0578
05D4 CA30 0001              60X̄/[N(N + 1)] → R45
05D8 41B0 36A0
05DC C860 0010              600X̄/[N(N + 1)] → R45
05E0 41B0 3590
05E4 4830 0578
05E8 CB30 0001
05EC 41B0 36A0              600X̄/[N(N + 1)(N - 1)] → R45
05F0 C860 0010
05F4 41B0 3590              6000X̄/[N(N + 1)(N - 1)] = 1000a → R45
05F8 08C4
05FA 08D5                   Save in RCD
05FC 0840
05FE 0851                   Recall ΣY_i to R45
0600 4830 0578
0604 41B0 36A0              ΣY_i/N → R45
0608 C860 1000
060C 41B0 3590              1000ΣY_i/N → R45
```

```
0610 0804              Save 1000 ΣY_i/N in R01
0612 0815
0614 4850 0578         Recall N to (R5)
0618 0200              Continue
061A CA50 0001         N + 1 → (R5)
061E CE50 0001         (N + 1)/2 → (R5)
0622 0865              Move (R5)to (R6)
0624 084C              Recall 1000a to (R45)
0626 085D
0628 41B0 3590         [(N + 1)/2][1000a] → (R45)
062C 0B15
062E 0F04              1000[ΣY_i/N - {(N + 1)/2}a] = 1000b → R01
0630 C8B0 0330
0634 030B              Return to main program
0636 0000
0638 0000
063A 0000
063C 0000   (0640-0658) = SET PARAMETERS TO READ STEP ROUTINE
063E 0000      [DAS]
0640 40B0 0656         Store return address
0644 C800 02FA         Load time per point for taking step data
0648 4000 3038              from (0646) to (3038)
064C C800 0400         Load number of points to be taken in step
0650 4000 302E              data from (064E) to (302E)
0654 C8B0 009E         Return
0658 030B
065A 40B0 0670 (065A-0672) = SET PARAMETERS TO READ ROUGHNESS DATA
065E C800 0004    [DAS]  Load time per point for taking roughness
0662 4000 3038              data from (0660) to (3038)
0666 C800 2000         Load number of points to be taken for roughness
066A 4000 302E              data from (0668) to (302E)
066E C8B0 01AA
0672 030B              Return
>
```

```
0A82 40B0 0AAC  (0A82-0AAE) = MULTIPLY (R34) BY KCAL^2, CONVERT TO DECIMAL
0A86 4860 0BAC     [ECT]      AND PRINT RESULT AT TTY
0A8A 4870 0BAE                Recall KCAL to (R67)
0A8E 41B0 34D4                Multiply by KCAL^2
0A92 41B0 34D4
0A96 C860 0000                Multiply by constant in (0A98.0A9C)
0A9A C870 FA00
0A9E 0853                     Move result to (R45)
0AA0 0842
0AA2 41B0 3300                Convert result to decimal
0AA6 41B0 32D0                Print result at TTY
0AAA C8B0 0CA2
0AAE 030B                     Return
0AB0 0000       (0AB2-0AF4) = MULTIPLY POWER OF 10 IN UNITS BY TWO AND
0AB2 40B0 0AF2     [ECT]      add number in (0ACC) TO RESULT
0AB6 C850 202D
0ABA 41B0 314A                Print space and - sign at TTY
0ABE D300 0BB1                Recall byte of (0BB0) containing power of
0AC2 41B0 3EA4                  ten convert ASCII to hexadecimal, result →
0AC6 CF10 0001                Multiple (R1) by 2                      (R1)
0ACA CA10 0003                Add (0ACC) to (R1)
0ACE 0851                     Move result to (R5)
0AD0 0744
0AD2 41B0 3300                Convert result to decimal, result → (R3)
0AD6 0823                     Move to (R2)
0AD8 41B0 3EF0                Print result at TTY
0ADC C850 5351
0AE0 41B0 314A                Print "SQ" at TTY
0AE4 41B0 3110                Print space at TTY
0AE8 C850 4D4D
0AEC 41B0 314A                Print "MM" at TTY
0AF0 C8B0 0CAE
0AF4 030B                     Return
0AF6 FFFF FFFF
0AFA FFFF FFFF
0AFE FFFF 0C0C  (0B00-0B04)  =  STORAGE FOR AVERAGE WAVELENGTH PARAMETERS
0B02 0036
0B04 EE80 FFFF  (0B0A-0B0C) = INDICES FOR TAKING AA DATA
0B08 FFFF 000C
0B0C 0000
0B0E 0E17          (0B0E)   = TEMPORARY STORAGE OF AA VALUE
0B10 0000
0B12 4120 0320  (0B14-0B5E) = STORAGE FOR AA VALUES FOR STATISTICAL
0B16 0325                       CALCULATIONS
0B18 0326
0B1A 0329
0B1C 0330
0B1E 032E
0B20 032F
0B22 0316
0B24 031B
0B26 030C
0B28 0311
0B2A 0319
0B2C 032E
0B2E 0312
0B30 0DAB
0B32 0DF8
```

```
0B34 0E0A
0B36 0DED
0B38 0DF9
0B3A 0E07
0B3C 0DFC
0B3E 0E09
0B40 0E15
0B42 0E29
0B44 0E1B
0B46 0E0B
0B48 0E00
0B4A 0DD9
0B4C 0DEA
0B4E 0DEC
0B50 0AAB
0B52 0A93
0B54 0009
0B56 0009
0B58 0009
0B5A 0009
0B5C 000A
0B5E 000A
0B60 0756        (0B60-0B82) = STORAGE FOR DOUBLE-PRECISION HEXADECIMAL
0B62 4840 0755                 VALUES OF H1, H2, ---, H8 and HM.
0B66 E540 0755
0B6A 8240 0755
0B6E 1F40 0754
0B72 BC40 0754
0B76 5940 0753
0B7A F640 0753
0B7E 9340
0B80 0754
0B82 EDC0 010E
0B86 0320
0B88 0116
0B8A 009A
0B8C 0118
0B8E 014D
0B90 11C2 1228   (0B90)       = A1, (0B92) = A2
0B94 1273 FFFF   (0B94)       = H0 VALUE
0B98 D9C0 FC3D   (0B96-8)     = a1
0B9C 3000 FFFF   (0B9A-C)     = b1
0BA0 C940 0375   (0B9E-0BA0)  = a2
0BA4 7000 FFFF   (0BA2-4)     = b2
0BA8 FFFF FFFF
0BAC 0001        (0BAC-E)     = KCAL
0BAE 8000 2D35   (0BB0)       = ASCII CODE UNITS INPUT AT TTY
0BB2 1111 FFFF   (0BB2)       = MAIN PROGRAM FLAG
0BB6 FFFF 40B0
0BBA 0BCE        (0BB8-0BD0) = PRINT, "AVG." AT TTY
0BBC C8A0 314A                 Load print subroutine address to (RA)
0BC0 C850 4156                 Load ASCII code for "AV" to (R5)
0BC4 01BA                      Print
0BC6 C850 472E                 Load ASCII code for "G." to (R5)
0BCA 01BA                      Print
0BCC C8B0 0EA6
0BD0 030B                      Return
0BD2 40B0 0BEE
```

```
0BD6 C8A0 314A  (0BD2-0BF0) = PRINT "SLP." AT TTY -
0BDA C850 534C     [ECT]
0BDE 01BA
0BE0 C850 502E
0BE4 01BA
0BE6 C850 3D20
0BEA 01BA
0BEC C8B0 0E74
0BF0 030B
0BF2 0000       (0C00-0CFE) = PROGRAM TO CALCULATE AND PLOT
0BF4 0000          [ECT]      THE AUTOCORRELATION FUNCTION OF DATA
0BF6 0000                     IN MEMORY LOCATIONS 1000 -3000 . THE
0BF8 0EFF                     512 SHIFT AUTOCORRELATION FUNCTION IS
0BFA FFFF C880                STORED IN MEMORY LOCATIONS 0680-0A80
0BFE 2000 41B0                CRL
0C02 3F44 0700                Clear (R0)
0C06 07CC                     Initialize the shift register
0C08 C8D0 0004                (0C06) = shift increment, here it equals
0C0C C8E0 0800                         twice the data point spacing.
0C10 C870 0002                Initialize summation index counter increment
0C14 C8F0 0001                Load panel display device number to (RF)
0C18 C840 000F                Load "F" to (R4)
0C1C 082C
0C1E CE20 0003                Write (turn on last 4 lights) at display
0C22 4280 0C2A                panel "F" when (RC) are an even multiple of
0C26 C840 0000                4, Write "0" when (RC) are an odd multiple
0C2A 9AF4                                                        of 4
0C2C C880 2000                Intialize summation index counter final value
0C30 C860 0002                Intialize summation index counter  initial value
0C34 0B8C                     2(N-S) → (R8) as new limit for summation index
0C36 082C                     S →(R2)
0C38 CA20 1000                Calculate i + s for start of shifted data
0C3C 4020 0C50                Store starting value of i + s at (0C50)
0C40 0733
0C42 0744                     Initialize accumulators; R345
0C44 0755
0C46 4896 1000                Bring data point into R9; [1000 + (R9)] →(R9)
0C4A CE90 0004                Shift (R9) right arithmetic 4 bits
0C4E 4816 109C                Bring shifted data point into R1;  (0C50)+(R6) →(R1)
0C52 CE10 0004                Shift (R1) right arithmetic 4 bits
0C56 0C09                     Multiply shifted and unshifted data; (R1) x (R9)→(R01)
0C58 0A51                     [Overflows if RMS > 1/3 Fullscale]
0C5A 0E40                     DP accumulate in R45
0C5C C160 0C46                i=N-S
                              Σ   y(i)y(i+s) = (N-S)ACF(s) → (R45)
0C60 0838                     i=1
0C62 CE30 0001                N-S → (R3)
0C66 41B0 36A0                [1/(N-S).] Σ y(i)y(i + s) = ACF(s) → (R45); S=0,2,4,-
0C6A CF40 000C                Shift (R4) left 12 bits arithmetic
0C6E CC50 0004                Shift (R5) right 4 bits logical
0C72 0A45                     (R5) + (R4) → (R4) = 1/16_10 ACF(s)
0C74 085C                     Adjust the shift increment to obtain proper storage
0C76 CE50 0001                   increment which should be 2
0C7A 4045 0680                (1/16_10) ACF(s) → [0680 + (R5)]
0C7E C1C0 0C18                Return to calculate next ACF(s) if s < limit
0C82 C8F0 0002                Load TTY device number to RF
0C86 41B0 3A00                Print "VARIANCE =" at TTY
0C8A 4830 0680                Recall (1/16_10) ACF(s) to (R3)
0C8E 0744
```

```
0C90 0711
0C92 4010 0A98
0C96 C810 FA00
0C9A 4010 0A9C
0C9E 41B0 0A82
0CA2 C810 0003
0CA6 4010 0ACC
0CAA 41B0 0AB2
0CAE 4820 0680
0CB2 0733
0CB4 4030 0CDA
0CB8 C420 7F80
0CBC 4330 0CD0
0CC0 CE20 0001
0CC4 CA30 0001
0CC8 4030 0CDA
0CCC 4300 0CB8
0CD0 C800 0680
0CD4 4000 3978
0CD8 C800 0003
0CDC 4000 397C
0CE0 C800 007B
0CE4 4000 3984
0CE8 4000 398C
0CEC C800 0084
0CF0 4000 3990
0CF4 C800 0400
0CF8 4000 399E
0CFC 4300 3950
0D00 0200
0D02 41B0 31F8
0D06 C8A0 314A
0D0A C850 204E
0D0E 01BA
0D10 C850 203D
0D14 01BA
0D16 41B0 30BA
0D1A 0842
0D1C 41B0 3648
0D20 4010 0D5E
0D24 0821
0D26 CB20 0001
0D2A 4020 0D90
0D2E CF10 0001
0D32 4010 0D52
0D36 4010 0D84
0D3A 0711
0D3C 0766
0D3E 0777
0D40 4831 0B14
0D44 41B0 3624
0D48 0A73
0D4A 0E62
0D4C CA10 0002
0D50 C510 001C
0D54 4280 0D40
0D58 0846
0D5A 0857
```

Load proper constant to KCAL routine
to account for 3 bit shift in KCAL and
4 bit shift in ACF calculation 8 x 8 x 16 = 1024;
$0.FA00_x = 1/1.024$
Multiply by KCAL and contant, convert to Decimal
and print at TTY
Increase power of ten in units by 3
Print at TTY; 2 x power of ten entered at units query
Begin calculation of number of shifts needed in
order to obtain 8 bit plotting accuracy
Clear accumulator for number of shifts
Since variance always + test top 8 bits
It top not zero continue shifting, otherwise branch
Shift (R2) right 1 bit to plot
Increment (R3) by 1
Accumulate shifts at (0CDA)
Continue shifting (R2)
(0CD0-0CD4) = Load and store starting address

Load and store amount data is to be shifted

Load and store maximum value to be plotted

Load and store bias of plot

Load and store 2X number of points to be plotted

Plot the ACF and return to monitor

(0D00-0DEC) = CALCULATE MEAN AND VARIANCE OF N AA VALUES

[ECT] Print "ENTER" at TTY
Load Print routine address to RA
Load ASCII code for space and N to R5
Print
Load ASCII code for space and = to R5
Print
Read 4 characters before "space" at TTY → R2
Move to (R4)
Convert to hexadecimal; results → (R1) = N
Store at (0D5E)

N-1 → (R2)
N-1 → (0D90)

2N → (0D52)
2N → (0D84)

Clear index and accumulator registers

Call AA value from [0B14 + (R1)] to R3
SP (R3) → DP (R23)

DP accumulate in (R67)

Loop statements

Move $\sum_{i-1}^{N} AA_i$ → (R45)

```
0D5C C830 000E                    N → (R3)
0D60 41B0 36A0                    Σ AA_i/N = AA → (R5) → (0B86)
0D64 4050 0B86
0D68 0711
0D6A 0766                         Reset index and accumulator registers
0D6C 0777
0D6E 4830 0B86                    Recall AA → (R3)
0D72 4B31 0B14                    AA - AA_i → (R3)
0D76 0853                         Put in R5 also
0D78 0C43                         (AA - AA_i)^2 → R45
0D7A 0A75
0D7C 0E64                         Σ(AA - AA_i)^2 → R67
0D7E CA10 0002
0D82 C510 001C                    Loop statements
0D86 4280 0D6E
0D8A 0857                         Move Σ(AA - AA_i)^2 to R45
0D8C 0846
0D8E C830 000D                    N-1 → (R3)
0D92 41B0 36A0                    [1/(N-1)] Σ (AA - AA_i)^2 ≡ VAR → R45
0D96 4200 0000
0D9A 4050 0B8A                    Store result at 0B8A
0D9E C440 7FFF                    If result not positive branch to "ERROR"
0DA2 4230 3A30
0DA6 41B0 3F44                    CRL
0DAA 41B0 39D4                    Print "MEAN AA =" at TTY
0DAE 4850 0B86                    Recall AA to (R5)
0DB2 41B0 0518                    Multiply by KCAL, convert to decimal and print at TTY
0DB6 41B0 3110                    Print space
0DBA 4850 0BB0                    Recall units to (R5)
0DBE 41B0 3A50                    Print units at TTY
0DC2 41B0 3F44                    CRL
0DC6 41B0 3A00                    Print "VARIANCE =" at TTY
0DCA 4830 0B8A                    Recall var to (R3)
0DCE 0744
0DD0 C810 0001                    Load constant to KCAL^2 routine
0DD4 4010 0A98
0DD8 0711
0DDA 4010 0A9C
0DDE 41B0 0A82                    Multiply VAR by KCAL^2, convert to decimal, print at
0DE2 0711                                                                       TTY
0DE4 4010 0ACC
0DE8 41B0 0AB2                    Print units and "SQ MM" at TTY
0DEC 4300 3C80                    Return to monitor
0DF0 41B0 3A50
0DF4 C850 5351  (0E00-0EE0)       = PROGRAM TO CALCULATE THE AVERAGE SLOPE AND
0DF8 41B0 314A    [ECT]             WAVELENGTH OF PROFILE IN MEMORY LOCATIONS 1000-3000
0DFC 4300 3C80
0E00 0722
0E02 C830 0002                    Initialize index counter, R2
0E06 C840 2000
0E0A 4862 1000                    Recall and divide [1000 + (R2)] by 2
0E0E CE60 0001
0E12 4062 1000                    Restore result to same location
0E16 C120 0E0A                    Repeat for all data from 1000 thru 3000
0E1A C820 2000
0E1E C830 FFFE                    Reinitialize R2 for a decrementing counter
0E22 C840 0000
0E26 4862 1000                    Call 1/2 y_n → (R6)
```

```
0E2A 4872 0FFE          Call 1/2 y(n-1) → (R7)
0E2E 0B67               Calculate 1/2 Δyn = 1/2 (yn - yn-1)
0E30 4062 1000          Store 1/2 Δyn at [1000 + (R2)]
0E34 C020 0E26          Repeat for all data from 3000x down to 1002x
0E38 41B0 04A0          Calculate AA of data from 1002 through 3000
0E3C 4050 0B00          Save result at 0B00
0E40 0722
0E42 0835               (0E40-0E50) = KCAL x AA of 1/2 Δyn → (R345)
0E44 0744
0E46 4860 0BAC
0E4A 4870 0BAE
0E4E 41B0 34D4
0E52 C860 0003          (0E52-0E5A) = (R34) x 2 x 10^5 → (R234)
0E56 C870 0D40
0E5A 41B0 34D4
0E5E 4030 0B02          Save result in (0B02-0B04) ≡ SLP
0E62 4040 0B04
0E66 41B0 3F44          CRL
0E6A 0200               Continue
0E6C 41B0 0BB8          Print "AVG."
0E70 41B0 0BD2          Print "SLP. ="
0E74 4840 0B02
0E78 4850 0B04          Recall SLP and divide by 930(10)
0E7C C830 03A2
0E80 41B0 36A0
0E84 41B0 3300          Convert result to decimal
0E88 41B0 32D0          Print result at TTY
0E8C 41B0 3110          Print space
0E90 4800 0BB0          Recall ASCII code for power of ten = -P
0E94 CB00 0001          Calculate -P-5 + 6 = -P + 1
0E98 0850
0E9A 41B0 314A          Print - P + 1 at TTY
0E9E 41B0 3F44          CRL
0EA2 41B0 0BB8          Print "AVG."
0EA6 41B0 0EE4          Print "WL. ="
0EAA 4840 0B14          Recall first AA value stored
0EAE 0755
0EB0 4830 0B00          (0B14)/(0B00) = 2 AA/Δy
0EB4 41B0 36A0
0EB8 0834
0EBA 0845               WL ≡ 2π Δx AA/Δy = π Δx (0B14)/(0B00)
0EBC C860 0B69
0EC0 C870 AE61          π x 930(10) = 2921.681168(10) = B69.AE61(16)
0EC4 41B0 34D4
0EC8 0842               Move WL to (R45)
0ECA 0853
0ECC 41B0 3300          Convert WL to decimal
0ED0 41B0 32D0          Print WL at TTY
0ED4 41B0 3110          Print Space
0ED8 C850 2D36          Load -6 to (R5)
0EDC 41B0 3A58          Print "-6 MM" at TTY
0EE0 4300 3C80          Return to monitor
0EE4 40B0 0EFA (0EE4-0EFC) = ROUTINE TO PRINT "WL. =" AT TTY
0EE8 C8A0 314A   [ECT]
0EEC C850 574C
0EF0 01BA
0EF2 C850 3D20
0EF6 01BA
```

```
0EF8 C8B0 1504
0EFC 030B
0EFE FFFF 4000 (0F00-0F3C)     = REGISTER STORAGE FOR ALL REGISTERS
0F02 0F4A          [ECT]              EXCEPT REGISTER B
0F04 4010 0F4E
0F08 4020 0F52
0F0C 4030 0F56
0F10 4040 0F5A
0F14 4050 0F5E
0F18 4060 0F62
0F1C 4070 0F66
0F20 4080 0F6A
0F24 4090 0F6E
0F28 40A0 0F72
0F2C 40C0 0F76
0F30 40D0 0F7A
0F34 40E0 0F7F
0F38 40F0 0F82
0F3C 030B
0F3E 0000
0F40 0000
0F42 0000
0F44 0000
0F46 0000      (0F48-0F84)     = RESTORE CONTENTS STORED BY 0F00
0F48 C800 0001   [ECT]
0F4C C810 0047
0F50 C820 FED7
0F54 C830 01FF
0F58 C840 0000
0F5C C850 000D
0F60 C860 0019
0F64 C870 4778
0F68 C880 0000
0F6C C890 0400
0F70 C8A0 314A
0F74 C8C0 0C0A
0F78 C8D0 0001
0F7C C8E0 001E
0F80 C8F0 0002
0F84 030B
0F86 0F6E
0F88 4000 0FC4 (0F88-0FB4)     = REGISTER STORAGE FOR REGISTERS
0F8C 4010 0FC8   [ECT]                0, 1, 2, 7, 8, 9, A, C, D, E, and F
0F90 4020 0FCC
0F94 4070 0FD0
0F98 4080 0FD4
0F9C 4090 0FD8
0FA0 40A0 0FDC
0FA4 40C0 0FE0
0FA8 40D0 0FE4
0FAC 40E0 0FE8
0FB0 40F0 0FEC
0FB4 030B
0FB6 0000
0FB8 0000
0FBA 0000
0FBC 0000
0FBE 0000
```

```
0FC0 0000
0FC2 C800 0000  (0FC2-0FEE)    = RESTORE CONTENTS STORED BY 0F88
0FC6 C810 0000     [ECT]
0FCA C820 1098
0FCE C870 0002
0FD2 C880 1F68
0FD6 C890 FFFE
0FDA C8A0 314A
0FDE C8C0 0098
0FE2 C8D0 0004
0FE6 C8E0 0800
0FEA C8F0 0001
0FEE 030B
0FF0 0000
0FF2 0000
0FF4 0000
0FF6 0000
0FF8 0000
0FFA 0000
0FFC 0000
0FFE 0000
1000 0958
>
```

```
1500 0700        (1500-1670) =  PROGRAM FOR MEASURING THE HEIGHT OF 3 CONSECUTIVE STEPS
1502 4000 2002     [ECT]        Initialize loop, counter = R0
1506 41B0 0640                  Set parameters for taking step data
150A 41B0 3010                  Wait for an interrupt and then take data
150E C850 C9CE                  Load "IN"
1512 41B0 314A                  Print
1516 41B0 3110                  Print SP
151A 41B0 31F8                  (151A-1528) = Print "Enter P1" + SP
151E C850 5031
1522 41B0 314A
1526 41B0 3110
152A 41B0 30BA                  Read 4 characters at TTY and store at (2000)
152E 4020 2000
1532 41B0 31F8                  (1532-1540) = Print "ENTER P2" + SP
1536 C850 5032
153A 41B0 314A
153E 41B0 3110
1542 41B0 30BA                  Read 4 characters at TTY and store at (0B90)
1546 4020 0B90
154A 41B0 31F8                  (154A-1558) = Print "ENTER P3" +SP
154E C850 5033
1552 41B0 314A
1556 41B0 3110
155A 41B0 30BA                  Read 4 characters at TTY and store at (0B92)
155E 4020 0B92
1562 41B0 31F8                  (1562-156C) = Print "ENTER P"
1566 C850 5020
156A 41B0 314A
156E 41B0 30BA                  Read and store at (2006)
1572 4020 2006
1576 4820 0B90                  (1576-157C) = Calculation of 2X number of data
157A 4820 2000                      Points for LSQ fit = P2 - P1
157E 4020 0562                  Store result at (0562)
1582 4820 2006
1586 4820 2000
158A 4020 2004                  P-P1 = Period of wave → (2004)
158E 4820 0562
1592 CE20 0001
1596 4020 0364                  Number of points → (0364)
159A 41B0 3F44                  CRL
159E 4820 0B92
15A2 4020 0590                  Recall and store starting address of LSQ fit
15A6 41B0 0560                  LSQ fit to data from (0B92) to (0B92) + (P2 - P1)
15AA 40C0 0B9E
15AE 40D0 0BA0
15B2 4000 0BA2                  (15AA-15B8) = Store a2(i) and b2(i)
15B6 4010 0BA4
15BA 4820 2000                  Recall and store new starting address for LSQ fit
15BE 4020 0590
15C2 41B0 0560                  LSQ fit to data from (2000) to (2000) + (P2 - P1)
15C6 40C0 0B96
15CA 40D0 0B98
15CE 4000 0B9A                  (15C6-1504) = Store a1(i) and b1(i)
15D2 4010 0B9C
15D6 41B0 0350                  Calculate step height in hexadecimal
15DA 4860 0BAC
15DE 4870 0BAE                  Recall KCAL to R67
15E2 0722
```

```
15E4 4830 0B30          Recall HM(I) to R234
15E8 4840 0B82
15EC 41B0 34D4          Multiply HM(i) by KCAL
15F0 C860 0008          Multiply result by 8 account for the 3 bit shift in
15F4 0777                     obtaining KCAL
15F6 41B0 34D4          Round-off result to double precision accuracy and move
15FA 0777                     result in R45
15FC CA40 8000
1600 0E37
1602 0E27
1604 0853
1606 0842
1608 41B0 3300          Convert result to decimal value and print value
160C 41B0 32D0
1610 41B0 3110          Print SP
1614 4820 2000
1618 4A20 2004
161C 4020 2000          P + I(P - P1) → (2000)
1620 4820 0B90
1624 4A20 2004
1628 4020 0B90          P2 + I(P - P1) → (0B90)
162C 4820 0B92
1630 4A20 2004
1634 4020 0B92          P3 + I(P - P3) → (0B92)
1638 4820 2002
163C CA20 0001
1640 4020 2002          (1638-164A) = Loop statements
1644 C520 0003
1648 4230 159E
164C 41B0 31D6          Print "MORE?"
1650 41B0 3E8A          Read one character at TTY
1654 C500 004E          Compare with a N
1658 4330 1664          Branch to exit if equal, otherwise continue
165C 41B0 3F44          CRL
1660 4300 1500          Return to start
1664 C810 0080          Reload usual no. of points for step routine to (0364)
1668 4010 0364
166C 4300 3C80          Return to monitor
1670 4080 D870
1674 D870 D8A0
1678 D8B0 D8C0
167C D8B0 D8A0
1680 D880 D870
>
```

```
                    (2500-25B2) =  PROGRAM FOR OBTAINING STATISTICS ON STEP HEIGHT PROGRAMS
                      [ECT]
2500 41B0 0640                     Set parameters for step height data input
2504 41B0 3010                     Wait for an interrupt and read data
2508 C850 C9CE                     Load "IN"
250C 41B0 314A                     Print
2510 C810 1000                     Load and store 1000 as H0
2514 4010 0B94
2518 C810 11C2                     Load (251A) and store as A1
251C 4010 0B90
2520 C810 1244                     Load (2522) and store as A2
2524 4010 0B92
2528 41B0 0300                     Calculate step height in hex
252C 41B0 0400                     Calculate KCAL assuming 1000 for H0
2530 4300 2548                     Branch to printing part of program
2534 41B0 3010                     Wait for an interrupt and read data
2538 C850 C9CE                     Load "IN"
253C 41B0 314A                     Print
2540 41B0 3F44                     CRL
2544 41B0 0300                     Calculate step height in hex
2548 C8A0 32D0                     Load print routine address to (RA)
254C 4820 0B96
2550 4830 0B98                     Load slope a1 to (R23)
2554 01BA                          Print (R23)
2556 41B0 3110                     Print SP
255A 4820 0B9A
255E 4830 0B9C                     Load intercept b1 to (R23)
2562 01BA                          Print
2564 41B0 3110                     Print SP
2568 4820 0B9E
256C 4830 0BA0                     Load Slope a2 to (R23)
2570 01BA                          Print
2572 41B0 3110                     Print SP
2576 4820 0BA2
257A 4830 0BA4                     Load intercept b2 to (R23)
257E 01BA                          Print
2580 41B0 3110                     Print SP
2584 4820 0BAC
2588 4830 0BAE                     Load KCAL to (R23)
258C 01BA                          Print
258E 41B0 3110                     Print SP
2592 4820 0B80
2596 4830 0B82                     Load HM (hex) to (R23)
259A 01BA                          Print
259C 41B0 3F44                     Print CRL
25A0 41B0 3240                     Print "H1---H3---H5---H7---HM"
25A4 41B0 3F44                     Print CRL
25A8 41B0 0440                     Convert hex steps to decimal and print
25AC 41B0 3F44                     Print CRL
25B0 4300 2534                     Repeat
25B4 0000
>
```

```
3010 40B0 301E (3010-3020) = ROUTINE TO WAIT FOR AN INTERRUPT, THEN READ DATA
3014 41B0 306E    [DAS]
3018 41B0 3022
301C C8B0 01B8
3020 030B        (3022-306C) = ROUTINE TO LOAD DATA FROM ANALOG TO DIGITAL CONVERTER
3022 40B0 306A    [DAS]              (ADC)
3026 07CC
3028 C8D0 0002                Initialize index counters with (302E) being number of
302C C8E0 2000                     data points to be loaded
3030 0766
3032 C870 0001                Initialize delay counter with delay being determined by
3036 C880 0004                     (3038)
303A C890 0000                ADC channel number → R9
303E C8F0 000A                Device number → RF
3042 DEF0 3064                Initializing command byte (3064) → ADC
3046 DEF0 3065                Enabling command byte (3065) → ADC
304A 9AF9                     ADC channel number to be read → ADC
304C DBFC 1000                Read one byte from ADC → [1000 + (RC)]
3050 DBFC 1001                Read one byte from ADC → [1001 + (RC)]
3054 C160 3054                Delay before reading next point
3058 0766                     Reset delay counter
305A C1C0 304A                Loop for all data
305E C8F0 0002                Restore TTY device number to RF
3062 4200 9086                Storage for device commands
3066 9F12                     Set condition codes (c.c.) and clear interrupt conditions
3068 C8B0 301C                Return to main program
306C 030B
306E 40B0 30AE (306E-30B8) = WAIT FOR AN INTERRUPT AT ADC MARK TIME FOR 1 SECOND
3072 C800 0000    [DAS]              THEN EXIT
3076 4000 0044                Load and store new program status word c.c.
307A C800 3096                Load and store new program status word
307E 4000 0046                     (address to jump to on interrupt)
3082 C8F0 000A                Load ADC device number to RF
3086 DEF0 30B2                Initialize interrupt at ADC (30B2) → ADC
308A DEF0 30B4                Enable interrupt at ADC (30B4) → ADC
308E C200 3092                Load current program status word to R0
3092 C000 3C80                Wait state:  Branch to monitor upon execute at console
3096 0200                     Continue
3098 DEF0 30B6                Send disable interrupt command to ADC
309C C8C0 0000
30A0 C8D0 0001                Set up counters for 1 second delay (30A6) determine
30A4 C8E0 2F00                     delay
30A8 C1C0 30A8                Mark time 1 second
30AC C8B0 3018                Return to main program
30B0 030B
30B2 1000 4000                Storage for device commands
30B6 8000 8000
30BA 40B0 30E2 (30BA-30E4) = ROUTINE TO READ 4 HEX CHARACTERS AT TTY INTO R2
30BE 41B0 3E8A    [DAS]       Read one character at TTY
30C2 C500 00A0                Compare with ASCII code for space
30C6 4330 30E0                Exit if equal
30CA C500 008D                Compare with ASCII code for CR
30CE 4330 30E0                Exit if equal
30D2 41B0 3EA4                Convert input character ASCII to hexadecimal
30D6 CD20 0004                Shift left logical (R2) 4 bits
30DA 0621                     Logical OR (R1) with (R2)
30DC 4300 30BE                Loop and read until SP or CR appears
30E0 C8B0 0106               (Last 4 characters read are left in R2)
```

```
30E4 030B       (30E8-310C) = ROUTINE TO READ TWO CHARACTERS AT TTY AND STORE ASCII
30E6 4220 40B0 [DAS & ECT]  CODE AT 0BB0
30EA 310A 41B0
30EE 3E8A 4200              Read one character into (R0 bits 8-15)
30F2 0000
30F4 0850                   Move to R5
30F6 CD50 0008              Shift contents of R5  8 bits left logical
30FA 41B0 3E8A              Read another character into (R0 bits 8-15)
30FE 4200 0000
3102 0A50                   Add to contents of R5
3104 4050 0BB0              Store at 0BB0
3108 C8B0 00D6
310C 030B                   Return
310E C8A0 40B0 (3110-3120)  = PRINT SINGLE SPACE ROUTINE
3112 311E C800 [DAS]
3116 00A0                   Put ASCII code for SP in (R0)
3118 41B0 3E80              Print character From bits 8-15 of (R0)
311C C8B0 3142              Return
3120 030B
3122 40B0 3144 (3122-3146) = PRINT 2N+1 SPACES ROUTINE
3126 C8C0 0001   [DAS]
312A C8D0 0001              Initialize index counter for number of spaces (3130) = N
312E C8E0 0003
3132 C850 A0A0              Put ASCII Code for 2 spaces in (R5)
3136 41B0 314A              Print at TTY (R5)
313A C1C0 3132              Loop for desired number of spaces.
313E 41B0 3110              Print SP
3142 C8B0 0216              Return to main program
3146 030B
3148 0824
314A 40B0 3168 (314A-316A) = PRINT TWO HEX CHARACTERS ASCII CODE IN (R5)
314E 0805        [DAS]      (R5) → (R0)
3150 C400 FF00              Mask off 8 bits on left of (R0)
3154 CC00 0008              Shift (R0) right 8 bits
3158 41B0 3E80              Print right byte of (R0) at TTY
315C 0805                   (R5) → (R0) again
315E C400 00FF              Mask off 8 bits on right of (R0)
3162 41B0 3E80              Print
3166 C8B0 0258              Return
316A 030B
316C 074A
316E 075A       (3170-318E) = ROUTINE TO PRINT "UNITS"
3170 40B0 318C   [DAS]
3174 C8A0 314A              Address for print two hex characters → (RA)
3178 C850 D5CE              ASCII code for UN → (R5)
317C 01BA                   Print
317E C850 C954              ASCII code for IT → (R5)
3182 01BA                   Print
3184 C850 D3A0              ASCII code for S space → (R5)
3188 01BA                   Print
318A C8B0 31A4              Return
318E 030B
3190 40B0 31AA (3190-31AC) = ROUTINE TO PRINT "AA ------- UNITS" + CRL
3194 C850 C1C1   [DAS]      ASCII code for AA → (R5)
3198 41B0 314A              Print
319C 41B0 3122              Print seven spaces
31A0 41B0 3170              Print "UNITS"
31A4 41B0 3F44              Print CRL
```

```
31A8 C8B0 01A6                Return
31AC 030B
31AE 2850 40B0 (31B0-31D4) = ROUTINE TO PRINT "H OR R?"
31B2 31D2 C8A0   [DAS]        Print routine address → (RA)
31B6 314A C850                ASCII code for H space → (R5)
31BA C8A0 01BA                Print
31BE C850 CFD2                ASCII code for OR → (R5)
31C2 01BA                     Print
31C4 C850 A0D2                ASCII code for space R → (R5)
31C8 01BA                     Print
31CA C850 3FA0                ASCII code for ? space → (R5)
31CE 01BA                     Print
31D0 C8B0 018A                Return
31D4 030B
31D6 40B0 31F2 (31D6-31F4) = ROUTINE TO PRINT "MORE?"
31DA C8A0 314A   [DAS]        Print routine address → (RA)
31DE C850 CDCF                ASCII code for MO → (R5)
31E2 01BA                     Print
31E4 C850 D2C5                ASCII code for RE → (R5)
31E8 01BA                     Print
31EA C850 3FA0                ASCII code for ? space → (R5)
31EE 01BA                     Print
31F0 C8B0 021A                Return
31F4 030B
31F6 3D62 40B0 (31F8-3216) = ROUTINE TO PRINT "ENTER"
31FA 3214 C8A0   [DAS]
31FE 314A C850
3202 C5CE 01BA
3206 C850 D4C5
320A 01BA
320C C850 D2A0
3210 01BA
3212 C8B0 00F6
3216 030B
3218 40B0 3234 (3218-3236) = ROUTINE TO PRINT "DATA"
321C C8A0 314A   [DAS]
3220 C850 C4C1
3224 01BA
3226 C850 D4C1
322A 01BA
322C C850 2FA0
3230 01BA
3232 C8B0 009A
3236 030B
3238 0C49
323A CD40 0001
323E CD50 40B0 (3240-327E) = ROUTINE TO PRINT "HI---H3---H5---H7---HM"
3242 327C C8A0   [ECT]
3246 314A C850
324A C8B1 01BA
324E 41B0 3122
3252 C850 C8B3
3256 01BA
3258 41B0 3122
325C C850 C8B5
3260 01BA
3262 41B0 3122
3266 C850 C8B7
```

```
326A 01BA
326C 41B0 3122
3270 C850 C8CD
3274 01BA
3276 41B0 3122
327A C8B0 0116
327E 030B
3280 A0D3 D4C5  (3280-32A2) = ASCII CODE FOR "STEP/ROUGHNESS CALIBRATION"
3284 D0AF D3D5
3288 D2C6 C1C3
328C C5A0 D2CF
3290 D5C7 C8CE
3294 C5D3 D3A0
3298 C3C1 CCC9
329C C2D2 C1D4
32A0 C9CF CEA0
32A4 40B0 32C8  (32A4-32CA) = ROUTINE TO PRINT "STEP/ROUGHNESS CALIBRATION" CENTERED
32A8 41B0 3122    [DAS]                ON PAGE
32AC C8C0 0000
32B0 C8DC 0002                 Initialize index counters
32B4 C8E0 0022
32B8 C8A0 314A                 Put print routine address in (RA)
32BC 485C 3280                 [3280 + (RC)] → R5
32C0 01BA                      Print
32C2 C1C0 32BC                 Loop for all characters
32C6 C8B0 008A                 return
32CA 030B
32CC CE40 0001  (32D0-32F0) = ROUTINE TO PRINT HEXADECIMAL FORM OF CONTENTS OF R2
32D0 40B0 32EE    [DAS]                AND R3
32D4 0851
32D6 0862                      Save (R2) and (R3) in R6 and R7
32D8 0873
32DA 0200
32DC 4200 0000
32E0 0826                      (R6) → (R2)
32E2 41B0 3EF0                 Print Hexadecimal form of (R2)
32E6 0827                      (R7) → (R2)
32E8 41B0 3EF0                 Print Hexadecimal form of (R2)
32EC C8B0 054E                 Return
32F0 030B
32F2 0E48
32F4 4300 2858
32F8 0000
32FA 0000
32FC 0000       (3300-33FE) = ROUTINE TO CONVERT BINARY NUMBER IN R45 TO DECIMAL IN
32FE 0D72         [CEK]                R123 [USES (R8) AS INDEX TO STORE RESULT IN
3300 40B0 33FC                         MMY STARTING AT ADDRESS IN (336A)]
3304 41B0 0F10                 Save registers 4 to F
3308 0788
330A 0200
330C 4200 0000
3310 4200 0000
3314 C8F0 8000                 Load 1 into 0 bit of RF
3318 04F4                      Logical and 0 bit of RF and R4 for sign
331A 4330 332E                 Branch if +
331E 0700                      Clear R0
3320 C740 FFFF                 Complement (R4) and (RF)
3324 C750 FFFF
```

```
3328 CA50 0001          Complete 2's complement
332C 0E40               Add carry to R4
332E 07CC               Clear RC
3330 C8D0 0004          Put increment in RD
3334 C8E0 0014          Put limit to get first 6 powers of 10 in RE
3338 0700               Clear R0
333A 0894               Save (R4) and (R5) in R9 and RA
333C 08A5
333E 4B4C 3530          Subtract power of 10,[3530 + (RC)] from (R4)
3342 4210 335C          Branch if result is negative
3346 0849               Restore R4
3348 4B5C 3532          Subtract power of 10,[3532 + (RC)] from (R4,5)
334C 4F4C 3530
3350 4210 335C          Branch if result is negative
3354 CA00 0001          Add 1 to (R0)
3358 4300 333A          Loop for more subtractions
335C 0849               Restore original contents to R4 and R5
335E 085A
3360 CCC0 0002          Divide (RC) by 4
3364 08BC               Put (RC) in RB
3366 0AB8               Index (RB) by (R8)
3368 D20B 3550          Store [R0(8:15)] at 3550 + (RB)
336C CDC0 0002          Multiply (RC) by 4
3370 C1C0 3338          Loop for next power of 10
3374 C8D0 0002          Reinitialize increment and limit for remaining power
3378 C8E0 001C                of 10
337C 0700               Clear R0
337E 08A5               Save R5
3380 4B5C 3530          Subtract power of 10,[3530 + (RC)], from (R5)
3384 4210 3390          Branch if result is negative
3388 CA00 0001          Add 1 to (R0)
338C 4300 337E          Loop for more subtractions
3390 085A               Restore original contents of R5
3392 CCC0 0001          Divide (RC) by 2
3396 08BC               Put (RC) into (RB)
3398 0AB8               Index (RB) by (R8)
339A D20B 354A          Store [R0(8:15)] at 354A + (RB)
339E CDC0 0001          Multiply (RC) by 2
33A2 C1C0 337C          Loop for next power of 10
33A6 D318 3550          Put 10**9 place value ([3550 + (R8)] → [R1(8:15)]) in R1
33AA CD10 0004          Shift left 1 digit (R1)
33AE D308 3551          Put 10**8 place value ([3551 + (R8)] → [R0(8:15)]) in R0
33B2 0610               Logical OR (R0) with (R1) into R1
33B4 061F               Logical OR sign bit with (R1) into R1
33B6 D328 3552          Put 10**7 place value ([3552 +(R8)] → [R2(8:15)]) in R2
33BA CD20 000C          Shift left (R2) 3 digits
33BE D308 3553          Put 10**6 place value ([3553 + (R8)] → [R0(8:15)]) in R0
33C2 CD00 0008          Shift left (R0) 2 digits
33C6 0620               Logical OR (R0) with (R2) into R2
33C8 D308 3554          Put 10**5 place value ([3554 + (R8)] → [R0(8:15)]) in R0
33CC CD00 0004          Shift left (R0) 1 digit
33D0 0620               OR result into (R2)
33D2 D308 3555          Put 10**4 place value, ([3555 + (R8)] → [R0(8:15)]) in R0
33D6 0620               OR result into (R2)
33D8 D338 3556          Put 10**3 place value, ([3556 + (R8)] → [R3(8:15)]) in R3
33DC CD30 000C          Shift left 3 digits
33E0 D308 3557          Put 10**2 place value ([3557 + (R8)] → [R0(8:15)]) in R0
33E4 CD00 0008          Shift left 2 digits
```

```
33E8 0630                       OR result into (R3)
33EA D308 3558                  10**|place value → [R0(8:15)]
33EE CD00 0004                  Shift left (R0)|digit
33F2 0630                       OR result into (R3)
33F4 0635                       OR (R5) = 10**0 place value into (R3)
33F6 41B0 0F58                  Restore registers 4 → F
33FA C8B0 0482                  Return to main program
33FE 0303
3400 40B0 3406
3404 4200 0710  (3408-34CE) = ROUTINE TO MULTIPLY UNSIGNED (R5) x (R9) → (R89)
3408 40B0 34CC    [DAS]
340C 4000 34A4
3410 4010 34A8                  Save all registers not used as input or output
3414 4020 34AC
3418 4030 34B0
341C 4040 34B4
3420 4060 34B8
3424 4070 34BC
3428 40A0 34C0
342C 40C0 34C4
3430 40D0 34C8
3434 0700                       Clear R0
3436 0815                       Shift (R5) to R1
3438 C410 0001                  Last bit of (R5) → (R1), logical AND operation
343C 07CC                       Clear RC
343E 08D1                       Shift (R1) to RD
3440 0722                       Clear R2
3442 0839                       Shift (R9) to R3
3444 C430 0001                  Last bit of (R9) → (R3), logical AND operation
3448 0CC3                       Last bit of (R5) x Last bit of (R9) → RD = P0
344A 0783                       Clear
344C CC90 0001                  Shift (R9) right 1 bit logical
3450 CC50 0001                  Shift (R5) right 1 bit logical
3454 0C09                       Last bit of (R5) x 1 bit shifted (R9) → R01 = P1
3456 0C25                       Last bit of (R9) x 1 bit shifted (R5) → R23 = P2
3458 0C65                       1 bit shifted (R9) x 1 bit shifted (R5) → R89 = P3
345A CD80 0CC2                  Shift (R8) 2 bits left logical
345E 0879                       Move contents of R9 → R7 (product result)
3460 C470 C000                  Pick off two high bits of right product result, P3
3464 CC70 000E                  Shift these two bits 14 bits to right
3468 0A87                       Add then to left part of product result P3
346A CD90 0002                  Shift right product result  2 bits left
346E CD00 0001                  Shift left product result of P1   1 bit left
3472 0871                       Move right product result of P1 to R9
3474 C470 8000                  Pick off high bit of right product result of P1
3478 CC70 000F                  Shift this bit 15 bits to right
347C 0A07                       Add to left product result of P1
347E CD10 0001                  Shift right product result of P1, 1 bit left
3482 0A91                       Add shifted right results of P3 and P1
3484 0E80                       Add with carry left results of P1 and P3
3486 CD20 0001                  Shift left result of P2 left 1 bit
348A 0873                       Move right result of P2 to R7
348C C470 8000                  Pick off high bit
3490 CC70 000F                  Shift it right 15 bits
3494 0A27                       Add it shifted left, left part of P2
3496 CD30 0001                  Shift right part of P2  1 bit left
349A 0A93                       Add shifted results of P2 and P3
349C 0E82
```

```
349E 0A9D                        Add result of P0 to get final result in R89
34A0 0E82
34A2 C800 FFFF
34A6 C810 FFF4                   Restore starting contents of all registers not used
34AA C820 0000                         for output or input
34AE C830 06E0
34B2 C840 0000
34B6 C860 0000
34BA C870 30C0
34BE C8A0 314A
34C2 C8C0 0000
34C6 C8D0 014F
34CA C8B0 3510                   Return to main program
34CE 030B
34D0 4230 3544 (34D4-3524) = ROUTINE TO MULTIPLY (R34) x (R67) → (R2345)
34D4 40B0 3522   [DAS]
34D8 41B0 0F18                   Store register 6-F
34DC 07CC
34DE 07DD
34E0 07EE                        Clear register C-F to accumulate result
34E2 07FF
34E4 0894                        Move (R4) and (R7) to R9 and R5
34E6 0857
34E8 41B0 3408                   (R4) x (R7) → (R89) = P0
34EC 08F9
34EE 08E8                        Move P0 to (REF)
34F0 0894
34F2 0856
34F4 41B0 3408                   (R4) x (R6) → (R89) = P1
34F8 0AE9                        P0 + P1 = P1
34FA 0ED8                                + P0
34FC 0893                               CD EF
34FE 0857
3500 41B0 3408                   (R3) x (R7) → (R89) = P2
3504 0AE9                                      P2
3506 0ED8                        P0 + P1 + P2 =  P1
3508 0893                                        P0
350A 0856                                     C D EF
350C 41B0 3408                   (R3) x (R7) → (R89) = P3
3510 0AD9
3512 0EC8                        Finish accumulating into RC
3514 082C
3516 083D                        Transfer result to (R2345)
3518 084E
351A 085F
351C 41B0 0F60                   Restore registers
3520 C8B0 0538
3524 030B                        Return
3526 0003
3528 0007
352A 0807
352C 0605
352E 0907        (3530-354C) = HEXADECIMAL EQUIVALENTS OF POWERS OF 10
3530 3B9A CA00                   (3530-3532) = 10**9
3534 05F5                        (3534-6) = 10**8
3536 E100 0098                   (3538-A) = 10**7
353A 9680                        (353C-E) = 10**6
353C 000F
```

```
353E 4240 0001                      (3540-2) = 10**5
3542 86A0 0000                      (3544-6) = 10**4
3546 2710 03E8                      (3548) = 10**3
354A 0064                           (354A) = 10**2
354C 000A                           (354C) = 10**1
354E 0000
3550 0000       (3550-3558) =       STORAGE AREA FOR HEX TO DECIMAL CONVERSION
3552 0000
3554 0000
3556 0502
3558 0100
355A 0000
355C 0000
355E 0000       (3560-358C) =       ROUTINE TO MULTIPLY R23.4 BY 2
3560 40B0 358A   [DAS]
3564 CF20 0001                      Shift (R2) 1 bit left arithmetic
3568 0853                           Move (R3)  R5
356A C450 8000                      Pick off high bit of R3, put in (R5)
356E CC50 000F                      Shift bit 15 bits right logical
3572 0A25                           Add it to shifted result in R2
3574 CD30 0001                      Shift (R3) 1 bit left logical
3578 0854
357A C450 8000                      Pick off high bit of (R4) and put in R5
357E CC50 000F                      Shift bit 15 bits right logical
3582 0A35                           Add it to shifted result in R3
3584 CD40 0001                      Shift (R4) 1 bit left logical
3588 C8B0 0000                      Return
358C 030B
358E CD20 40B0  (3590-35C0) =       ROUTINE TO MULTIPLY (R45) BY (R6)→R345
3592 35BE 0200   [ECT & DAS]        (R45) EITHER SIGN, (R6) POSITIVE
3596 41B0 35C4                      |(R45)|→(R45)
359A 0777
359C 0788
359E 0896                           Move (R6) → R9
35A0 41B0 3408                      (R5) x (R6) → (R89) = P0
35A4 07AA                           Clear RA
35A6 08B4                           Move (R4) → RB
35A8 0CA6                           (R4) x (B6) → (RAB) = P1
35AA 0A8B                                          P1
35AC 0E7A                           P0 + P1  =    +P0
35AE 0837                                        789
35B0 0848                           Move result to (R345)
35B2 0859
35B4 41B0 35F8                      Change sign if original (R45) negative
35B8 41B0 0FC2
35BC C8B0 03B2                      Return
35C0 030B
35C2 0002       (35C4-35F4) =       |(R45)| → R45 ABSOLUTE VALUE ROUTINE SET FLAG AT
35C4 40B0 35F2   [ECT & DAS]              35EE IF NEGATIVE
35C8 41B0 0F88                      Save register 0,1,2,7-F
35CC 07AA                           Clear RA
35CE 40A0 35EE                      Clear Flag Location
35D2 0844                           Load (R4) to R4 to test sign
35D4 4210 35DC                      Branch if negative
35D8 4300 35E8                      Exit
35DC C8A0 8000                      Load flag to (RA)
35E0 40A0 35EE                      Store Flag
35E4 41B0 35F8                      Change sign of (R45)
```

```
35E8 41B0 0FC2                Restore original contents of R0,1,2,7-F
35EC 4200 8000                (35EE) = Flag storage
35F0 C8B0 359C                Return
35F4 030B
35F6 0000       (35F8-3620) = ROUTINE TO CHANGE SIGN OF (R45) IF FLAG, 8000, AT
35F8 40B0 361E [ECT & DAS]          35EE IS SET
35FC 48A0 35EE                Load Flag from (35EE)
3600 4330 361C                Exit if no flag
3604 C740 FFFF                Take 2's complement of (R45)
3608 C750 FFFF
360C CA50 0001
3610 4280 3618                If carry bit set add 1 to (R4)
3614 4300 361C                If not exit
3618 CA40 0001
361C C8B0 35B8                Return to program
3620 030B
3622 C1C0 40B0 (3622-3642) = ROUTINE TO CONVERT SINGLE PRECISION (R3) TO DOUBLE
3626 3640 0823   [DAS]              PRECISION IN (R23)
362A CE30 0000                (3628) = Put (R3) into R2
362E C420 8000                Test sign bit of (R3)
3632 4230 363A                If bit present i.e. (R3) are negative, Branch otherwise
3636 4300 363E                     exit
363A C820 FFFF                Load FFFF into (R2)
363E C8B0 382E                Return
3642 030B
3644 40B0 32FE (3648-3696) = ROUTINE TO CONVERT DECIMAL (R4), 4 DIGITS, TO
3648 40B0 3694 [NS & DAS]           HEXADECIMAL AT R1
364C 0700
364E 0711                     Clear (R0) and (R1)
3650 C820 000A                Load A(=10) to (R2)
3654 0834                     Shift (R4) → R3
3656 C430 F000                Pick off  high 4 bits or first digit of (R4)
365A 4330 3664                If zero branch to test next lower digit
365E CC30 000C                If not zero shift (R3) right 12 bits
3662 0A13                     Add shifted (R3) to (R1)
3664 0C02                     (R1) X A → (R01)
3666 0834                     Recall (R4) again
3668 C430 0F00                Pick off next digit of (R4)
366C 4330 3676                If zero branch
3670 CC30 0008                If not shift (R3) right 8 bits
3674 0A13                     Add shifted (R3) to (R1)
3676 0C02                     (R1 X A → (R01)
3678 0834                     Recall (R4) again
367A C430 00F0                Pick off next digit of (R4)
367E 4330 3688                If zero, branch
3682 CC30 0004                If not, shift (R3) right 4 bits
3686 0A13                     Add shifted (R3) to (R1)
3688 0C02                     (R1) X A → (R01)
368A 0834                     Load (R4) → R3
368C C430 000F                Pick off last digit (last 4 bits) of (R4)
3690 0A13                     Add to (R1)
3692 C8B0 040C
3696 030B                     Return to main program
3698 4280 3656
369C 4300 3C80 (36A0-374C) = ROUTINE TO DIVIDE 2 HEX HALFWORDS BY 1 HEX HALFWORD;
36A0 40B0 374A [CEK]                (R45) ÷ (R3) → (R45)
36A4 41B0 0F88
36A8 4060 3746                Save all registers not used for input or output
```

```
36AC CC50 0001          Shift (R5) right logical 1 bit
36B0 CE40 0001          Shift (R4) right arithmetic 1 bit
36B4 4280 36BC          If a least significant bit shifted out of (R4) branch
36B8 4300 36C0          If not branch around high bit add to (R5)
36BC CA50 8000          Add high bit to (R5)
36C0 0788               Clear R8
36C2 C890 0001          Load "1" to (R9)
36C6 C8A0 FFFF          Load "FFFF" to (RA)
36CA 0814               Load high order part to R1 to test sign
36CC 4310 36D8          Branch if (R45) are positive
36D0 074A               Two's complement (R45)
36D2 075A
36D4 0A59
36D6 0E48
36D8 0825               Save low-order part in R2
36DA CC50 000F          Shift (R5) right logical 15 bits
36DE CF40 0001          Shift (R4) left 1 bit arithmetic
36E2 0654               OR (R4) into (R5) High Order 15 bits
36E4 CE40 000F          Shift (R4) right 15 bits arithmetic
36E8 0D43               (R45)/(R3) → (R5) Remainder → (R4)
36EA 0805               Save result in R0, High Order 15 bits of answer
36EC 0852               Reload low-order part of input to (R5)
36EE C450 7FFF          Pick off low 15 bits of (R5)
36F2 CE40 0001          Remainder/2 → (R4)
36F6 4380 36FE          If no bit shifted out branch to next divide
36FA C650 8000          If bit shifted out OR high bit into (R5)
36FE 0D43               Divide present (R45) by (R3), (R45) = low order 15 bits
3700 0824               2nd remainder → (R2)                    of answer
3702 0840               Recall high order 15 bits of answer to (R4)
3704 CE40 0001          Shift (R4) right 1 bit arithmetic
3708 4380 3710          Branch if no bit shifted out
370C C650 8000          If bit shifted out OR high bit into (R5)
3710 CF20 0001          2nd remainder times 2
3714 0523               Is 2nd remainder times 2 > or = to divisor
3716 4280 371E          Branch if not
371A 0A59               Add 1 with carry to (R45)
371C 0E48
371E 0811               Was number negative
3720 4310 372C          Branch if number is positive
3724 074A
3726 075A               Complement (R45)
3728 0A59
372A 0E48
372C CF40 0001          Multiply (R45) by 2 to reverse
3730 CD50 0001          Process at start of this routine
3734 4280 373C
3738 4300 3740
373C CA40 0001
3740 41B0 0FC2          Restore all registers not used for input or output
3744 C860 0020
3748 C8B0 03EE
374C 030B               Return to main program
374E 0000
3750 0000
3752 0000
3754 0000
3756 0000
3758 0000
```

(3750-37E2) = PROGRAM FOR TAKING LEFT JUSTIFIED 12 BIT BINARY NOS. FROM MEMORY LOCATIONS 1000 TO 3000, CONVERTING THEM TO SIGNED FOUR DIGIT DECIMAL NOS., AND PRINTING (AND PUNCHING) IN A FORMAT OF NUMBER-SPACE WITH TEN NUMBERS PER LINE. PLUS SIGNS ARE NOT PRINTED IN ORDER TO BE COMPATIBLE WITH THE NBS 1108 CENTRAL COMPUTER FACILITY.

```
>3750R
>37E4P          [ECT & CY]
3750 41B0 3F26                  Punch a leader for tape output
3754 07EE                       Clear line and word index counters
3756 07AA
3758 483E 1000                  [1000 + (RE)] → (R3) Load data to (R3)
375C CE30 0004                  Shift (R3) right 4 bits arithmetic
3760 41B0 3624                  Single precision (R3) → double precision (R23)
3764 0853
3766 0842                       Move to R45
3768 41B0 3300                  Convert (R45) to decimal in (R123)
376C 41B0 37A0                  Print and punch 4 digits and sign
3770 41B0 3110                  Print SP
3774 CAE0 0002                  Index (RE) by 2
3778 CAA0 0001                  Index (RA) by 1
377C C5E0 2000                  Compare (RE) with word limit
3780 4280 3788                  If (RE) > 2000 go to 3788, otherwise continue
3784 4300 3C80                  Return to monitor
3788 C5A0 000A                  Compare (RA) with limit of words per line
378C 4280 3758                  If (RA) < A go to 3758, otherwise continue
3790 41B0 3F44                  CRL
3794 C5E0 2000                  Check word limit again
3798 4280 3756                  If (RE) < 2000 take next data entry
379C 4300 3C80                  Return to monitor
37A0 40B0 37EC (37A0-37E2) =    PRINT AND PUNCH ROUTINE
37A4 41BC 0F10                  Save contents of registers A-F
37A8 0852                       Move (R23) to R56
37AA 0863
37AC 0821                       Test sign of data point
37AE C410 F000
37B2 4330 37BE                  If + branch, If - continue
37B6 C800 002D                  Load ASCII code for - to (R0)
37BA 4300 37C2                  Branch to print and punch contents of R0
37BE C800 00A0                  Load ASCII code for + to (R0)
37C2 41B0 3E80                  Print (R0) at TTY
37C6 C420 0FFF
37CA 4200 0000                  Printing (R1) and (R2) bypassed since 12 bits
37CE 0825                           corresponds to only 4 digits
37D0 4200 0000
37D4 0826                       Load (R3) into R2
37D6 41B0 3EF0                  Print (R2)
37DA 41B0 0F58                  Restore (RA to RF)
37DE C8B0 3770
37E2 030B                       Return to program
>
```

```
                    (3800-3888) =  PROGRAM TO CALCULATE THE AMPLITUDE OR PROBABILITY
                      [ECT]        DENSITY FUNCTION FOR DATA STORED AT (1000-3000),
                                   STORES RESULT IN (0680-0A80), AND PLOTS FUNCTION
 3800R                             AT STRIP CHART RECORDER
>3888
3800 07CC                          3800 - First entry to initialize storage area at
3802 07AA                                 0680-0A80.
3804 40AC 0680
3808 CAC0 0002
380C C5C0 0400
3810 4280 3804
3814 0777                          3814 - Second entry to generate amplified function
3816 C880 0002
381A C890 2000                     Initialize index counter (R7)
381E C860 0008                     8 → (R6)
3822 4837 1000                     [1000 +(R7)] → (R3)
3826 CE30 0004                     Shift (R3) right 4 bits arithmetic
382A 41B0 3624                     Single precision (R3) → double precision (R23)
382E 0D26                          Calculate box data point should belong  to
3830 0822                          Load remainder to test if > 0
3832 4310 383A                     If remainder < 0, add -1, then add bias
3836 CA30 FFFF                     If remainder ≥ 0, just add bias
383A CA30 0100                     Add bias
383E CF30 0001                     Multiply box number by 2 to obtain address no. of
3842 48C3 0680                     [0680 + (R3)] → (RC)                          storage
3846 CAC0 0001                     Add 1 to (RC)
384A 40C3 0680                     New contents of RC → [0680 + (R3)]
384E C170 3822                     Increment index counter then repeat
3852 C880 0680
3856 4080 3978
385A C880 0000                     (3852-3888) = Initialization of plot routine, plot
385E 4080 397C                           (0680-0A80), then return to monitor.
3862 C880 0002
3866 4080 399A
386A C880 0400
386E 4080 399E
3872 C880 00F0
3876 4080 3984
387A 4080 398C
387E C880 000F
3882 4080 3990
3886 4300 3950
>
```

```
3900 07AA        (3900-3916) = PROGRAM TO CLEAR CONTENTS OF MEMORY LOCATIONS 1000 to
3902 07CC          [ECT]                   3000
3904 40AC 1000
3908 CAC0 0002
390C C5C0 2000
3910 4280 3904
3914 4300 3C80
3918 C800 1000 (3918-3946) = PROGRAM TO INITIALIZE PLOT ROUTINE FOR BIPOLAR NUMBERS
391C 4000 3978   [ECT]          AND TO PLOT DATA FROM MEMORY LOCATIONS 1000 TO 3000
3920 C800 0008                  ON STRIP CHART RECORDER
3924 4000 397C
3928 C800 007B
392C 4000 3984
3930 4000 398C
3934 C800 0084
3938 4000 3990
393C C800 2000
3940 4000 399E
3944 4300 3950 (3950-39CF) = PLOT ROUTINE:  THIS ROUTINE TAKES HALFWORDS FROM
3948 0000          [ECT]        SPECIFIED MEMORY LOCATIONS USING THE 8 MOST SIGNIFICANT
394A 0000                       BITS (INCLUDING THE SIGN BIT) AND BIASES THE DATA TO
394C 0000                       HAVE A RANGE FROM 4 TO 255.  SEE NOTES IN TEXT.
394E 0000
3950 C800 0003                  Turn on recorder
3954 41B0 3E80
3958 4200 0000
395C C800 0004                  Print low value
3960 41B0 3E80
3964 41B0 39B6                  Continue for ∿ 8 sec.
3968 C800 00FF
396C 41B0 3E80                  Print high value
3970 41B0 39B6                  Continue for ∿ 8 sec.
3974 0799                       Initialize index counter (R9)
3976 4839 0680                  [(3978)+(R9)] → (R3)
397A CE30 0000                  Shift (R3) right arithmetic proper number of bits
397E 4320 398E                  If number is positive, continue: otherwise branch
3982 C530 00F0                  Compare number with maximum value
3986 4280 398E                  If less than maximum add bias
398A C830 00F0                  If greater than or equal max, replace no. with maximum
398E CA30 000F                  Add bias
3992 0803
3994 41B0 3E80                  Plot biased value
3998 CA90 0002
399C C590 0400                  Increment index counter, compare with limit, continue
39A0 4280 3976                      if (R9) < limit.
39A4 C800 0002
39A8 41B0 3E80                  Turn off recorder
39AC 4300 3C80                  Return to monitor
39B0 0000
39B2 0000
39B4 0000
39B6 40B0 39CC (39B6-39CF) = WAIT APPROXIMATELY 8 SEC. ROUTINE.
39BA C8C0 0000
39BE C8D0 0001
39C2 C8E0 FFF0
39C6 C1C0 39C6
39CA C8B0 3974
39CE 030B
```

```
39D0 0000
39D2 0000
39D4 40B0 39FC  (39D4-39FE) = ROUTINE TO PRINT AT TTY "MEAN AA ="
39D8 C8A0 314A    [ECT]
39DC C850 4D45
39E0 01BA
39E2 C850 414E
39E6 01BA
39E8 C850 2041
39EC 01BA
39EE C850 4120
39F2 01BA
39F4 C850 3D20
39F8 01BA
39FA C8B0 1704
39FE 030B
3A00 40B0 3A28  (3A00-3A2A) = ROUTINE TO PRINT AT TTY "STD. DEV.="
3A04 C8A0 314A    [ECT]
3A08 C850 5354
3A0C 01BA
3A0E C850 4420
3A12 01BA
3A14 C850 4445
3A18 01BA
3A1A C850 5620
3A1E 01BA
3A20 C850 3D20
3A24 01BA
3A26 C8B0 1704
3A2A 030B
3A2C 3A2C 3A2E
3A30 C8A0 314A  (3A30-3A48) = PROGRAM TO PRINT "ERROR" THEN RETURN TO MONITOR
3A34 C850 4552    [ECT]
3A38 01BA
3A3A C850 524F
3A3E 01BA
3A40 C850 5220
3A44 01BA
3A46 4300 3C80
3A4A 3A4A 3A4C
3A4E 3A4E 40B0  (3A50-3A70) = ROUTINE TO PRINT POWER OF TEN ENTERED AT UNITS QUERY
3A52 3A6E CA50    [ECT]       FOR STEPS AND SAME + (-1) FOR ROUGHNESS AND ALSO TO
3A56 0001                     PRINT "MM".
3A58 40B0 3A6E
3A5C 41B0 314A
3A60 41B0 3110
3A64 C850 4D4D
3A68 41B0 314A
3A6C C8B0 01D0
3A70 030B
3A72 3A72 3A74
3A76 3A76 3A78
3A7A 3A7A 3A7C
3A7E 3A7E 4000
>
```

```
3A40 C850 5220                  Load R space
3A44 01BA                       Print
3A46 4300 3C80                  Return to monitor
3A4A 3A4A 3A4C
3A4E 3A4E 40B0  (3A50-3A70) = PRINT MM AND INCREASE POWER OF 10 BY 1 IF FOR AA
3A52 3A6E CA50                  (REPEAT OF PREVIOUS ITEM)
3A56 0001
3A58 40B0 3A6E
3A5C 41B0 314A                  Print power of 10
3A60 41B0 3110                  Print SP
3A64 C850 4D4D                  Load MM
3A68 41B0 314A                  Print
3A6C C8B0 01D0                  Return to main program
3A70 030B
3A72 3A72 3A74                  REMAINDER OF PROGRAMS AND ROUTINES WRITTEN
3A76 3A76 3A78                  BY PHILIP G. STEIN
3A7A 3A7A 3A7C  (3A80-3AD7) = ILLEGAL INSTRUCTION HANDLER
3A7E 3A7E 4000
3A82 3AB6 4010                  Save 0
3A86 3ABA 4020                  Save 1
3A8A 3ABE 4030                  Save 2
3A8E 3AC2 4080                  Save 3
3A92 3AC6 40B0                  Save 8
3A96 3ACA 40F0                  Save B
3A9A 3ACE C8F0                  Save F
3A9E 0002                       "2" to F
3AA0 C800 003F                  Load ?
3AA4 41B0 3E80                  Print
3AA8 41B0 3F44                  CR-LF
3AAC 4820 0032                  Load old PSW address
3AE0 4120 3EF0                  Print 4
3AB4 C800 008A                  Restore 0
3AB8 C810 004B                  Restore 1
3ABC C820 E000                  Restore 2
3AC0 C830 0006                  Restore 3
3AC4 C880 0008                  Restore 8
3AC8 C8B0 3F58                  Restore B
3ACC C8F0 0002                  Restore F
3AD0 C200 3AD4                  Load PSW
3AD4 8000 3C80                  PSW - Wait State - to monitor
3AD8 0700       (3AD8-3AE9) = INITIALIZE ILLEGAL INSTRUCTION INTERRUPT WITH COMMAND
3ADA 4000 0034                        (&)
3ADE C800 3A80
3AE2 4000 0036
3AE6 4300 3C80
3AEA 0000       (3B00-3B4B) = ANNOTATION PROGRAM
3AEC 0000       THIS PROGRAM WILL ACCEPT A MONITOR PRINT COMMAND IN THE NORMAL
3AEE 0000       FORMAT.  IT WILL PRINT ONE LINE, THEN TAB OVER TO THE COMMENTS FIELD.
3AF0 0004       AFTER TYPING COMMENTS, A RUBOUT WILL GET THE NEXT LINE OF TEXT.  A
3AF2 0630       CARRIAGE RETURN WILL GET A LINE FEED AND ROOM FOR MORE COMMENTS.
3AF4 0635
3AF6 4300 285A
3AFA 0000                       TO ACTIVATE ANNOTATOR COMMAND IS (T)
3AFC 0000                       TO STOP ANNOTATOR COMMAND IS (U)
3AFE 0000                       MONITOR CONTENTS DURING ANNOTATION AT 3DA8 ARE CHANGED.
3B00 C800 41B0                  To mess up print routine
3B04 4000 3DA8                  Store
3B08 4300 3C8A                  To monitor
```

```
3B0C 0000        (3B10-3B1A) =  STOP ANNOTATOR
3B0E 0000
3B10 C800 4200                  No-op to fix print routine
3B14 4000 3DA8                  Store
3B18 4300 3C8A                  To monitor
3B1C 0000        (3B20-3B4A) =  ANNOTATOR
3B1E 0000
3B20 C800 0089                  Load Tab
3B24 41B0 3E80                  Print
3B28 41B0 3E8A                  Ask for input
3B2C C500 008D                  Compare with CR
3B30 4330 3B40                  If equal, new line
3B34 C500 00FF                  Compare with rubout
3B38 4330 3DAC                  If equal, back to print routine
3B3C 4300 3B28                  Otherwise, loop for another character
3B40 C800 000A                  Load LF
3B44 41B0 3E80                  Print
3B48 4300 3B20                  Jump to tab
3B4C 0000        (3B50-3B7D) =  PRINT 32 CHARACTERS FROM MMY STORED IN ASCII STARTING
3B4E 0000                           AT ADDRESS IN (R3)
3B50 40B0 3B7A                  Save return
3B54 4030 3B5C                  Store First address in load instruction
3B58 07CC                       Clear index
3B5A D30C 008E                  Load character
3B5E 0800                       Test
3B60 4330 3B74                  If zero (BLANK) exit
3B64 41B0 3E80                  Print
3B68 CAC0 0001                  Increment index
3B6C C5C0 0020                  Compare with limit
3B70 4280 3B5A                  Loop if less than limit
3B74 41B0 3F44                  CR-LF
3B78 C8B0 2B4A                  Restore return
3B7C 030B        (3B80-3BDD) =  ROUTINE FOR PUNCHING AT TTY HEXADECIMAL FORM OF MMY
3B7E 0000                       CONTENTS FROM ADDRESS IN (R5) TO (R4)
3B80 C8A0 3E80                  Address for punch subroutine
3B84 0700                       Clear 0
3B86 41B0 3F26                  Punch blank leader
3B8A C8C0 3F50                  Start address of loader
3B8E C8E0 3FCF                  Final address of loader
3B92 D30C 0000                  Fetch byte of loader
3B96 01BA                       Punch
3B98 C1C0 3B92                  Loop for next byte
3B9C C8C0 0001                  Prepare to punch blank spaces
3BA0 C8E0 0010
3BA4 0700
3BA6 01BA                       Punch
3BA8 C1C0 3BA6                  Loop
3BAC C800 00FF                  Rubout
3BB0 01BA                       Punch
3BB2 0815                       Start Address to I
3BB4 41C0 3BDE                  Punch
3BB8 0814                       Final address to I
3BBA 41C0 3BDE                  Punch
3BBE 0799                       Clear 9 for Hashsum
3BC0 08C5                       Start address to C
3BC2 08E4                       Final address to E
3BC4 D30C 0000                  Fetch byte of text
3BC8 0790                       Update Hashsum
```

```
3BCA 01BA                          Punch
3BCC C1C0 3BC4                     Loop for next byte
3BD0 0809                          All text punched. Hashsum to 0
3BD2 01BA                          Punch
3BD4 0700                          Clear 0
3BD6 41B0 3F26                     Punch blank leader
3BDA 4300 3C8A  (3BDE-3BEB) = ROUTINE TO PUNCH ADDRESS FROM (R1)
3BDE 0801                          Address into 0
3BE0 CC00 0008                     Shift 0 right 8
3BE4 01BA                          Punch top half
3BE6 9210                          Bottom half to 0
3BE8 01BA                          Punch
3BEA 030C                          Exit
3BEC 0000
3BEE 0000
3BF0 0000
3BF2 0000
3BF4 0000
3BF6 0000
3BF8 0000
3BFA 0000       (3C00-3C3F) = ROUTINE TO READ 32 CHARACTERS AT TTY AND STORE IN
3BFC 0000                          ASCII FORM IN MMY STARTING AT THE ADDRESS IN (R3)
3BFE 0000
3C00 40B0 3C3C                     Save return
3C04 4030 3C10                     Store first address in store instruction
3C08 C8C0 001E                     Index
3C0C 07DD                          Clear D
3C0E 40DC 008E                     Clear data space
3C12 CBC0 0002                     Subtract 2
3C16 4310 3C0E                     BR. if positive or zero
3C1A 4030 3C2C                     Store First address in store instruction
3C1E 41B0 3B8A                     Read one character
3C22 C500 008D                     CR?
3C26 4330 3C3A                     If so, return
3C2A D20D 008E                     Store Character
3C2E CAD0 0001                     Increment Index
3C32 C5D0 0020                     Compare with limit
3C36 4280 3C1E                     Loop if less than limit
3C3A C8B0 2B72                     Restore Return
3C3E 030B                          Exit
3C40 0000       (3C46-3C7F) = ROUTINE FOR PRINTING AT TTY HEXADECIMAL FORM CONTENTS
3C42 0000                          OF MMY FROM ADDRESS IN (R5) TO ADDRESS IN (R4)
3C44 0000
3C46 41B0 3F44                     CR-LF
3C4A 0825                          Address pointer from 5 into 2
3C4C 41B0 3EF0                     Call Print 4
3C50 C800 00A0                     Load SP
3C54 41B0 3E80                     Print. Gives double space after address
3C58 C8EC 0007                     (7) + C into E.  Contents of C are not used
3C5C C800 00A0                     Load SP
3C60 41B0 3E80                     Print
3C64 4825 0000                     Fetch Byte From Address In 5
3C68 41B0 3EF0                     Print 4
3C6C 0A5F                          Increment 5
3C6E C1C0 3C5C                     Loop for next Byte
3C72 0554                          Check for print limit
3C74 4380 3C8A                     If limit, go to dispatcher
3C78 41B0 3EE0                     If not, check panel button 15
```

```
3C7C 4330 3C46                 If button not pushed, print another line; otherwise
3C80 C800 0020 (SEE 3D04)        continue to this statement which is entrance to
3C84 C8F0 0002                 TTY device number                              monitor
3C88 9EF0                      Output command
3C8A 41B0 3F44                 CR-LF
3C8E C800 00BE                 Load >
3C92 41B0 3E80                 Print
3C96 41B0 3E8A                 Get character from TTY
3C9A 07CC                      Clear C - start index
3C9C C8D0 0001                 Increment
3CA0 C8E0 001E                 Last index
3CA4 D31C 3DE0                 Fetch byte from command table
3CA8 0510                      Compare with input byte
3CAA 4330 3CB6                 BR. if equal
3CAE C0C0 3CC0                 BR. if end of table reached
3CB2 4300 3CA4                 Loop for next character
3CB6 CDC0 0001                 Command found! Shift C left 1 to get index
3CBA 48BC 3E10                 Fetch address from jump table into B
3CBE 030B                      Jump indirect through B
3CC0 41B0 3EA4                 No command found. was data. go ASCII-Hex
3CC4 CD40 0004                 Shift 4 left 4
3CC8 0641                      Or new data in
3CCA 4300 3C96                 Go back for next character
3CCE 0000          (3CD0-3D03) = PRINT N/2 HALFWORDS STARTING WITH ADDR. IN (R3)
3CD0 40B0 3D00                 Save return
3CD4 4030 3CDC                 Store AX2 in Load instruction
3CD8 07CC                      Clear C
3CDA 482C 3868                 Load AX2(C) in 2
3CDE C8F0 0002                 "2" to F
3CE2 41B0 3EF0                 Print 4
3CE6 C800 00A0                 Load SP
3CEA 41B0 3E80                 Print
3CEE CAC0 0002                 Increment index
3CF2 C5C0 0006                 Compare with Limit, (3CF4) = N
3CF6 4280 3CDA                 Loop if less than limit
3CFA 41B0 3E80                 Print 2nd SP
3CFE C8B0 3850                 Restore return
3D02 030B                      Exit
3D04 00C0          (3C80-3CCD) = COMMAND INTERPRETER. ACCEPTS DATA INPUT FROM THE TTY
3D06 0000          AND BUFFERS IT IN R4.  ACCEPTS COMMANDS FROM TTY AND CALLS
3D08 0000          APPROPRIATE ROUTINE TO SERVICE THEM.
3D0A 0000
3D0C 0000          (3D10-3DD5) = SERVICE ROUTINES FOR R, RO, I, CR, SP, G, P AND W
3D0E 0000                              COMMANDS FROM TTY
3D10 0854          *R SERVICE*   Load 4 into 5, thereby setting address pointer
3D12 0744                        Clear 4 (Suppress leading zeroes on typein)
3D14 4300 3C8A                   Jump to dispatcher
3D18 0B5F          *RUBOUT SERVICE*  Decrement pointer by 2
3D1A C800 00DE                   Load Arrow
3D1E 41B0 3E80                   Print
3D22 4300 3D12                   Jump to clear 4 and return
3D26 0A5F          *I SERVICE*   Increment pointer by 2
3D28 4300 3D12                   Jump to clear 4 and return
3D2C 4045 0000 *CR SERVICE*      Store 4 at location in 5
3D30 0A5F                        Increment pointer by 2
3D32 4300 3D12                   Jump to clear 4 and return
3D36 4045 0000 *SP SERVICE*      Store 4 at location in 5
3D3A 0A5F                        Increment pointer by 2
```

```
3D3C 0744                          Clear 4
3D3E 4300 3C96                     Return to dispatcher without >
3D42 C540 3B80  *G SERVICE*        Is address > monitor start?
3D46 4280 3D52                     If it is, jump
3D4A C540 3000                     Address > monitor end?
3D4E 4280 3C96                     If so, jump to dispatcher without >
3D52 41B0 3F44                     If not, call CR-LF
3D56 C800 0087                     Load bell
3D5A 41B0 3E80                     Print
3D5E 41B0 3EE0                     Check Panel Button 15
3D62 4330 3D5E                     Loop waiting for button
3D66 0304                          Jump to address in 4
3D68 41B0 3F44  *P SERVICES*       Call CR-LF
3D6C 0825                          Address pointer into R2 to be printed
3D6E 41B0 3EF0                     Print 4
3D72 C800 00A0                     Load SP
3D76 41B0 3E80                     Print
3D7A 4825 0000                     Fetch halfword at pointer into R2
3D7E 0812                          Also into R1
3D80 C410 F000                     Strip top 4 bits for opcode check
3D84 4330 3D9E                     Jump if 0. was halfword instruction
3D88 C510 9000                     Compare to 9
3D8C 4330 3D9E                     Jump if 9. was halfword instruction
3D90 41B0 3EF0                     Print 4
3D94 0A5F                          Increment pointer
3D96 C800 00A0                     Load SP
3D9A 41B0 3E80                     Print
3D9E 4825 0000                     Fetch halfword at pointer into R2
3DA2 41B0 3EF0                     Print 4
3DA6 0A5F                          Increment pointer
3DA8 4200 3B20                     To annotator (no-operation normally)
3DAC 0554                          Have we reached print limit?
3DAE 4380 3DB6                     If not continue, otherwise branch to print last line
3DB2 41B0 3EE0                     If not, is panel button 15 pushed
3DB6 4330 3D68                     If not, get next line of print
3DBA 4300 3C8A                     If button pushed or at print limit, return to dispatcher
3DBE 0200       *W SERVICE*
3DC0 41B0 3F44                     CR-LF
3DC4 0825                          (R5) → R2
3DC6 41B0 3EF0                     Print 4 (current address pointer)
3DCA C800 00A0                     Load SP
3DCE 41B0 3E80                     Print
3DD2 4300 3C96                     Return to dispatcher without >
3DD6 0000
3DD8 0000
3DDA 0000
3DDC 0000
3DDE 0000       (3DE0-3E0F) = STORAGE FOR MONITOR COMMAND TABLE
3DE0 8DFF A050                     CR, R0, SP, P
3DE4 D247 C90A                     R, G, I, LF
3DE8 BED4 5355                     >, T, S, U
3DEC 4BD7 CC00                     K, W, L, BLANK
3DF0 48CA 4D56                     H, J, M, V
3DF4 D1D8 A6A9                     Q, X, &, )
3DF8 2859 5A4E                     (, Y, Z, N
3DFC CF21 A300                     O, !, #
3E00 0000
3E02 0000
```

```
3E04 0000
3E06 0000
3E08 0000
3E0A 0000                       (Modified by ECT for use by surface measurement programs)
3E0C 0000
3E0E 0000       (3E10-3E6F) =   JUMP TABLE FOR COMMANDS (SEE 3CA4-3CBE)
3E10 3D2C 3D18                  CR, RO
3E14 3D36 3D68                  SP, P
3E18 3D10 3D42                  R, G            J - Amplitude density function
3E1C 3D26 3C8A                  I, LF           V - Autocorrelation function
3E20 3C80 3B00                  >, T            H - Steps
3E24 0260                       S               Q - Roughness
3E26 3B10 3B80                  U, K            S - Calibration
3E2A 3DC0 3C46                  W, L            Y - Wavelength
3E2E 3C96 017E                  BLANK, H        M - Mean and Standard Deviation
3E32 3800 0D00                  J, M            X - Clear (1000-3000)
3E36 0C00                       V               Z - Plot (1000-3000)
3E38 019E                       Q
3E3A 3900 3AD8                  X, &
3E3E 3C8A 3C8A                  ),(
3E42 0E00                       Y
3E44 3918 3C8A                  Z, N
3E48 3C8A 3C8A                  0, !
3E4C 3C8A 3C8A                  #
3E50 3C8A 3C8A
3E54 3C8A 3C8A
3E58 3C8A 3C8A
3E5C 3C8A 3C8A
3E60 3C8A 3C8A
3E64 3C8A 3C8A
3E68 3C8A 3C8A
3E6C 3C8A 3C8A
3E70 0000
3E72 4200 0000
3E76 4200 0000
3E7A 0000
3E7C 0000
3E7E 0000
3E80 4300 3F60                  PRINT ONE CHARACTER MOVED TO 3F60, JUMP THERE
3E84 0000
3E86 0000       (3E8A-3E9A) =   READ ONE CHARACTER, WAITS FOR DATA AT TTY, PUTS INPUT
3E88 0000                           IN RO
3E8A C880 00A4                  Waits for data from TTY, puts it in RO
3E8E 9EF8                       Unblock, Read, Disable
3E90 9DF8                       Sense Status
3E92 0888                       Check, Status, If zero, data is ready
3E94 4230 3E90                  Loop if not zero
3E98 9BF0                       Read data
3E9A 030B                       Exit
3E9C 0000
3E9E 0000
3EA0 0000       (3EA4-3EC5) =   CONVERT ASCII BYTE IN RO TO HEX IN RI
3EA2 0000
3EA4 0810                       Input byte to RI
3EA6 C410 0070                  Pick off bits 9, 10, 11
3EAA C510 0040                  Are they 1, 0, 0?
3EAE 4330 3EBA                  If so, Letter between A and O, BR
3EB2 0810                       If not, its a number.  Bring in byte again.
```

```
3EB4 C410 000F                  Bits 12 thru 15 in 1. This is the answer.
3EB8 030B                       Exit
3EBA 0810                       Letter. Bring in byte again
3EBC CA10 0009                  Add 9 (Since a comes in as "41")
3EC0 C410 000F                  Bits 12 thru 15 in 1. This is the answer.
3EC4 030B                       Exit
3EC6 0000       (3EC8-3EDF) =   CONVERT HEX CHAR, IN R0 TO ASCII BYTE IN R1
3EC8 0810                       Pick up char, put in 1, subtract 9
3ECA CB10 0009                  If neg, number between 0 and 9
3ECE 4220 3EDA                  Append 'B0' to make ASCII
3ED2 C600 00B0
3ED6 0810                       Put in 1 as output.
3ED8 030B                       Exit
3EDA C610 00C0                  Number was between A and F. Take subtracted value
3EDE 030B                       Append 'C0' and exit
3EE0 C8F0 0001                  '1' to F (Panel Dev. No.)
3EE4 9BF0                       Read panel
3EE6 C8F0 0002                  '2' to F
3EEA C400 0001                  Check for last bit zero (Button 15) sets condition code
3EEE 030B                       Exit
3EF0 0733       *PRINT HEX HALFWORD*   Clear 3
3EF2 40B0 3F20                  Save return
3EF6 0802                       Pick up input argument from 2
3EF8 C400 F000                  Strip off bits 0, 1, 2, 3
3EFC CC00 000C                  Shift right 12
3F00 41B0 3EC8                  Hex-ASCII
3F04 0801                       Result in 0
3F06 41B0 3E80                  Print
3F0A 0803                       Get 3 into 0
3F0C CB00 0006                  Subtract 6 (Count Limit)
3F10 4310 3F1E                  If exceeded, exit
3F14 0A3F                       Otherwise, increment 3 by 2
3F16 CD20 0004                  Shift left 4 to get second character
3F1A 4300 3EF6                  Loop back to print more
3F1E C3B0 3D94                  Restore Return
3F22 0303                       Exit
3F24 0000       (3F26-3F41) =   LEADER GENERATOR FOR TAPE PUNCHING
3F26 40B0 3F3E                  Store return
3F2A 07CC                       Clear C
3F2C C8D0 0001                  Put '1' in D
3F30 C8E0 0048                  Put '48'in E
3F34 41B0 3E80                  Print
3F38 C1C0 3F34                  Loop and count
3F3C C8B0 3BDA                  Restore return
3F40 030B                       Exit
3F42 0000       (3F44-3F5D) =   CARRIAGE RETURN-LINE FEED SERVICE
3F44 40B0 3F5A                  Save return
3F48 C800 008D                  Load CR
3F4C 41B0 3E80                  Print
3F50 C800 008A                  Load LF
3F54 41B0 3E80                  Print
3F58 C8B0 3D6C                  Restore Exit
3F5C 030B                       Exit
3F5E 0000       (3F60-3FCF) =   PRINT 1 CHARACTER FROM (R0)
3F60 C880 00A8                  Unblock, write, disable
3F64 9EF8                       Output command
3F66 9DF8                       Sense status
3F68 4280 3F60                  Loop on TTY busy
```

```
3F6C 9AF0                       Write data
3F6E 030B                       Exit
3F70 0000
3F72 0000
3F74 0000
3F76 0000
3F78 0000 (3F80-3FBI) = BOOTSTRAP LOADER -- READS STARTING AND FINAL BYTE ADDRESSES
3F7A 0000 FROM TAPE.  DOES NOT LOAD HASHSUM OR FOLLOWING BLANKS.  RETURNS TO MONITOR.
3F7C 0000 NOTE THAT '50' LOADER LEAVES '1' IN 3, '2' IN A, AND '94' IN 0079
3F7E 0000
3F80 C850 3FB2                  Address for fetch byte
3F84 01B5                       Get A byte
3F86 C500 00FF                  Is it a rubout?
3F8A 4230 3F84                  No. Keep looking
3F8E 41C0 3FC2                  Rubout found. Address to 4
3F92 0824                       4 to 2. Starting byte address
3F94 41C0 3FC2                  Final byte address to 4
3F98 0799                       Clear 9 for hashsum
3F9A 01B5                       Get next byte
3F9C 0790                       Include in hashsum
3F9E 9A39                       Hashsum to Panel Lights
3FA0 D202 0000                  Store byte where 2 points
3FA4 C120 3F9A                  Br. on index < or =
3FA8 01B5                       Get hashsum
3FAA 0790                       Update hashsum
3FAC 9A39                       Hashsum to panel lights
3FAE 4300 3C80                  To monitor
3FB2 DEA0 0079 *GET BYTE FROM TTY*  Output command. Block, read, disable
3FB6 9DA8                       Sense status
3FB8 0888                       Test
3FBA 4230 3FB6                  Loop until data ready
3FBE 92A0                       Read the data
3FC0 030B                       Exit
3FC2 01B5       *FETCH ADDRESS INTO R4*  Get next byte
3FC4 0840                       Put it in 4
3FC6 CD40 0008                  Shift 4 left 8. Top half of address
3FCA 01B5                       Get Next byte.  Bottom half of address
3FCC 9204                       Put it in 4
3FCE 030C                       Exit
3FD0 3FD0 3FD2
3FD4 3FD4 3FD6
3FD8 3FD8 3FDA
3FDC 3FDC 3FDE
3FE0 3FE0 3FE2
3FE4 3FE4 3FE6
3FE8 3FE8 3FEA
3FEC 3FEC 3FEE
3FF0 3FF0 3FF2
3FF4 3FF4 3FF6
3FF8 3FF8 3FFA
3FFC 3FFC FFFF
4000 0000
>
```

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBSIR 75-924	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

Evaluation, Revision and Application of the NBS Stylus/Computer System for Surface Roughness Measurement: Minicomputer Software

5. Publication Date April 1975

6. Performing Organization Code

7. AUTHOR(S) E. Clayton Teague

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No. 2131114

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address (Street, City, State, ZIP)

same

13. Type of Report & Period Covered Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

A thorough description of all the software used at NBS for characterizing surface texture is given in this report. The description includes flow diagrams and detailed, annotated listings of machine language programs for step-calibrating the system, for acquiring digitized surface profiles and for calculating from these profiles important parameters and statistical functions. Parameters and functions included are the arithmetic average value, mean square value, average wavelength, average slope, amplitude density function and autocorrelation function.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Minicomputer software; Machine language; Linear least-squares fit; Arithmetic average; Amplitude density function; Autocorrelation function; Surface texture measurement.

18. AVAILABILITY [XX] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[ ] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13

[XX] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT) UNCLASSIFIED

21. NO. OF PAGES 78

20. SECURITY CLASS (THIS PAGE) UNCLASSIFIED

22. Price $ 5.00